



# Realizability of schedules by stochastic time Petri nets with blocking semantics

Loïc Hélouët, Karim Kecir

## ► To cite this version:

Loïc Hélouët, Karim Kecir. Realizability of schedules by stochastic time Petri nets with blocking semantics. Science of Computer Programming, 2018, 157, pp.71-102. 10.1016/j.scico.2017.12.004 . hal-01942241

**HAL Id: hal-01942241**

**<https://inria.hal.science/hal-01942241>**

Submitted on 3 Dec 2018

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics.

Loïc Hélouët

*INRIA Rennes, loic.helouet@inria.fr*

Karim Kecir

*Alstom, karim.kecir@alstom.com*

---

## Abstract

This paper considers realizability of expected schedules by production systems with concurrent tasks, bounded resources that have to be shared among tasks, and random behaviors and durations. Schedules are high level views of desired executions of systems represented as partial orders decorated with timing constraints. Production systems (production cells, train networks...) are modeled as stochastic time Petri nets STPNs with an elementary (1-bounded) semantics. We first propose a notion of time processes to give a partial order semantics to STPNs. We then consider *boolean realizability*: a schedule  $S$  is realizable by a net  $\mathcal{N}$  if  $S$  embeds in a time process of  $\mathcal{N}$  that satisfies all its constraints. However, with continuous time domains, the probability of a time process with exact dates is null. We hence consider *probabilistic realizability* up to  $\alpha$  time units, that holds if the probability that  $\mathcal{N}$  realizes  $S$  with constraints enlarged by  $\alpha$  is strictly positive. Upon a sensible restriction guaranteeing time progress, boolean and probabilistic realizability of a schedule can be checked on the finite set of symbolic prefixes extracted from a bounded unfolding of the net. We give a construction technique for these prefixes and show that they represent all time processes of a net occurring up to a given maximal date. We then show how to verify existence of an embedding and compute the probability of its realization.

*Keywords:* Petri nets, unfolding, scheduling, realizability.

---

## 1. Introduction

Correct scheduling of basic operations in automated systems (manufacturing or transport systems...) is a way to manage at best available resources, avoid undesired configurations, or achieve an objective within a bounded delay. Following a predetermined schedule is also a way to meet QoS objectives. For instance, operating a metro network or a fleet of buses usually amounts to implementing at best a predetermined timetable to meet users needs. In many cases, schedules are designed in order to avoid congestion problems, and ease up recovery from minor

delays that are part of the normal behavior of the system. Failing to implement  
10 a chosen schedule may then result in a loss of QoS, or even lead to congestion  
when delays accumulate. Schedules are high-level views for correct ordering of  
important operations in a system. They consider time issues such as delays  
between tasks, optimal dates, durations... in a production plan. They can be  
seen as partial orders among basic tasks, that abstract low-level implementation  
15 details, and are decorated with dates and timing constraints.

Designing a correct and optimal schedule for a system is a complex problem.  
On one hand, occurrence dates of events can be seen as variables, and correct  
and optimal schedules as optimal solutions (w.r.t. some criteria) for a set  
of constraints over these variables. Linear programming solutions have been  
20 proposed to optimize scheduling in train networks [1, 2]. On the other hand,  
optimal solutions (for instance, w.r.t. completion date) are not necessarily the  
most probable nor the most robust ones: indeed, systems such as metro networks  
are subject to small delays (variations in trips durations from a station to another,  
passengers misbehavior...). Delays are hence expected and considered as part of  
25 the normal behavior of the system. To overcome this problem, metro schedules  
integrate small recovery margins that avoid the network performance to collapse  
as soon as a train is late. Consequently, optimal and realizable schedules are not  
necessarily robust enough if they impose tight realization dates to systems that  
are subject to random variations (delays in productions, faults...). Note also  
30 that the size of timetabling problems for real systems running for hours cannot  
be handled in a completely automated way by tools, that usually have to be  
guided by experts to return quasi-optimal solutions. Hence, timetables design  
involves expert competences that differ from those needed to design systems.

When a schedule and a low-level system description are designed separately,  
35 nothing guarantees that the system is able to realize the expected schedule.  
This calls for tools to check realizability of a schedule. One can expect systems  
designers and timetable experts to share a common understanding of the behavior  
of their system, so in general, the answer to a boolean realizability question  
should be positive. However, being able to realize a schedule does not mean  
40 that the probability to meet optimal objectives is high enough. Obviously, in  
systems with random delays, the probability to realize a schedule with precise  
dates is null. Beyond boolean realizability, a schedule shall hence be considered  
as *realizable* if it can be realized up to some bounded imprecision, and with a  
significant probability.

45 This paper addresses realizability of schedules by stochastic timed systems.  
We define schedules as labeled partial orders decorated with dates and timing  
constraints, and represent systems with elementary stochastic time Petri nets  
(STPNs for short), a model inspired from [3]. We particularly emphasize on  
resources: non-availability of a resource (represented by a place) may *block*  
50 transitions. This leads to the definition of a *blocking semantics* for STPNs that  
forbids firing a transition if one of its output places is filled. We give a formal  
operational semantics for STPNs, and show that this semantics can also be  
captured in a concurrent setting by *time processes*. We then propose a notion  
of *realizability*: a schedule  $S$  is *realizable* by an STPN  $\mathcal{N}$  if  $S$  embeds in a time

55 process of  $\mathcal{N}$  that meets constraints on dates and causal dependencies of events in  $S$ . We prove that upon some reasonable time progress assumption, realizability can be checked on a finite set of *symbolic processes*, obtained from a bounded untimed unfolding [4, 5] of  $\mathcal{N}$ . Symbolic processes are processes of the unfolding with satisfiable constraints on occurrence dates of events. A symbolic framework  
60 to unfold time Petri nets was already proposed in [6, 7] but blocking semantics brings additional constraints on firing dates of transitions. Embedding of a schedule in a process of  $\mathcal{N}$  only guarantees *boolean realizability*: the probability of a time process in which at least one event is forced to occur at a precise date is null. We use transient analysis techniques for STPNs as proposed in [3] to  
65 compute the probability that a schedule is realized by a symbolic time process of  $\mathcal{N}$  up to an imprecision of  $\alpha$ . This allows to show that  $\mathcal{N}$  realizes  $S \pm \alpha$  with strictly positive probability, and then define a notion of *probabilistic realizability*.

The paper is organized as follows: Section 2 gives the main motivations for the models used in the paper, namely stochastic time Petri nets with a blocking  
70 semantics, and schedules. Section 3 gives an interleaved and a concurrent semantics to STPNs. Section 4 defines a notion of unfolding and symbolic process for STPNs. Section 5 shows how to verify that a schedule is compatible with at least one process of the system and measure the probability of such realization. Due to lack of space, proofs and several technical details are omitted,  
75 but can be found in Appendices.

## 2. Motivations

This section gives motivations for the models that are used in this paper. The context for this study is the modeling of automated systems with concurrently running tasks. In particular the models proposed hereafter can address *production*  
80 *systems* with a broad understanding of this term, ranging from automated production cells, to transport networks and human organizations. A typical example of such systems where concurrency is prominent is *urban train systems*, which motivated many features of our model, and in particular an unusual blocking semantics for stochastic time Petri nets.

85 A second topic addressed in this paper is accuracy of schedules. We want to ensure that a system with random perturbations can follow a predetermined schedule up to some bounded imprecision. Schedules are objects of everyday's life, and are both used as documentation for a system (for instance, bus and train schedules indicate to users when and where to pick up their transport), and to  
90 guide a system along behaviors that guarantee some quality of service. Though schedules are often perceived as linear orderings of actions, some scheduled events are causally related, and some other are independent (think for instance of tasks in a Gantt diagram, or buses moving in different parts of a city). Natural models for schedules should hence consider partial ordering.

### 95 2.1. A Petri net variant for urban train systems.

First of all, let us justify why concurrency models are preferable in the systems considered in this paper. Even in the case of finite resources, an

optimized production system runs several subtasks concurrently. Consider for instance a manufacturing systems that transform objects with tools: each object is eventually produced according to a production flow. Objects productions are independent most of the time, except when a common resource needs to be used. This calls for the use of non-interleaved representation of processes. The same consideration applies for train systems: a train can continue its planned trip as long as the track ahead of it is not occupied by another vehicle. Considering a bi-directional network of  $n$  stations with  $p$  trains, one can rapidly have to consider more than  $\binom{n}{k}$  states, which is not needed, as the decision for a train to move is mainly a local decision, that can be taken without knowing the status of the whole system (i.e. without maintaining a global state of the system). A non-interleaved model for these systems is hence preferable if the targeted applications do not call for calculus of global states of the system. In this paper, we will show that the targeted application (ensuring realizability of schedules) can be done with a concurrent semantics.

Urban train systems are usually composed of closed imbricated loops. Trains travel at predetermined speeds following a predetermined itinerary. They move from a station to another following a track. A network is hence not just a simple cycle: it contains forks, junctions, etc. Nevertheless, complex topologies can easily be captured by the flow relations of a Petri net. Consider for instance the example network of Figure 1. This network is composed of 7 stations and bidirectional tracks. A train in the network can be scheduled to serve repeatedly stations  $A.B.C.D.E.F.G.\bar{F}.\bar{E}.\bar{D}.\bar{C}.\bar{B}.\bar{A}$ , or follow smaller loops  $A.B.C.D.\bar{D}.\bar{C}.\bar{B}.\bar{A}$  and  $D.E.F.G.\bar{G}.\bar{F}.\bar{E}.\bar{D}$ . The Petri net at the bottom part of Figure 1 can represent this network: places are used to represent stations or tracks between stations, and transitions model a possible move from a part of the network to a consecutive one. The flow relation of this net is almost isomorphic to the original network.

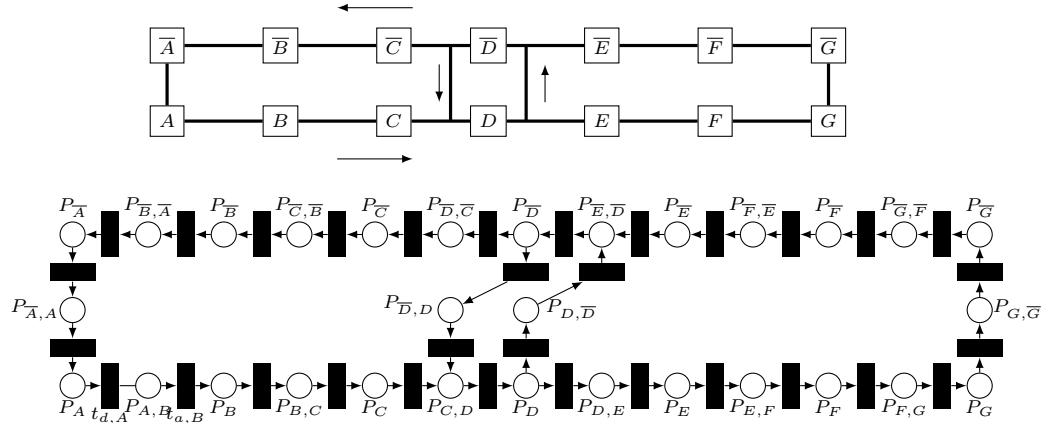


Figure 1: Modeling trains flows in a network with Petri nets

Petri nets are good models to represent workflows (e.g., to show how trains

can move in a network), but are of little use without timing information and randomness. A frequent question asked for train networks is whether the service is properly ensured. This means in particular whether trains serve station at a regular enough pace, or in case departure dates of trains are very sparse, whether trains arrivals and departures match an expected timetable.

**Time:** The second notion that a model for train systems must address is time. Several models of time have been proposed for timed concurrency models, and especially for Petri nets. Without claiming for exhaustiveness, one can cite at least time Petri nets [8] (TPNs for short), timed Petri nets (see for instance [9]), and time Petri nets with multiple enablings (see for instance [10]). Time Petri nets associate a rational time interval  $I(t) = [l, u]$  or  $I(t) = [l, \infty)$  to every transition  $t$  in the system. The semantics of time Petri nets considers that a clock is attached to every transition, and reset every time the transition becomes enabled. A transition can fire only if its clock's value lays within the interval  $I(t)$ . A particularity of this model is that time is not allowed to progress when a clock  $x_t$  reaches the upper bound of interval  $I(t)$ . This phenomenon is called *urgency*, and allows to model mandatory limits for execution dates. This is of particular interest to handle requirements such as: "A train has to leave at most 10 seconds after a departure order was given". Extensions of time Petri nets with multi-server semantics have been proposed to handle several enablings (instances) of each transition. In this paper, we consider a setting where trains are isolated from one another via a so-called fixed-block policy: track portions are reserved for one particular train. This prevents train from being too close from one another. Though this is not the only nor the most efficient way to avoid collision, this mechanisms is used in real systems <sup>1</sup>. We discuss this issue further in details in this section. Nevertheless, with such a fixed block policy, one can address a train network at block level (i.e., attach a place to each portion of the network that can be entered by at most one train), and hence multi-enabling needs not be considered.

The second well-known timed variant of Petri nets is called timed Petri nets. In this model, markings associate a set of real values to places, and flow relations from places to transitions are constrained by intervals. Tokens are hence not blind tokens as in TPNs but rather ages: a newly created token has age 0, and a transition  $t$  can fire iff every place  $p$  in its preset contains a token satisfying the constraint attached to  $p$  and  $t$ . A drawback of this model is that transition firing is not urgent: a transition that can fire is not forced to fire, and once tokens in a place become too old to satisfy a constraint allowing them to leave the place, they can be forgotten (or not considered). A natural assumption is that trains are modeled as tokens. However, discarding tokens amounts to losing trains, which is an undesirable feature for the systems we consider. Hence, in the context of train systems, urgency seems to be an unavoidable feature. Solution to integrate urgency in timed Petri nets have been proposed [11]. Though there must be way to avoid losses in timed Petri nets with adequate extensions, we

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Railway\\_signalling#Fixed\\_block](https://en.wikipedia.org/wiki/Railway_signalling#Fixed_block)

will not follow this approach in this paper and use a model with urgency.

**Randomness:** As soon as one is interested in performances of train systems, and not only in worst case scenarios, modeling randomness of these systems is essential. Indeed, train systems are subject to many random events: mechanical failures, bad weather conditions, passengers misconduct,... Some of these random events have huge impact on the network performance (hours of delay), but are usually rare. Recovering from such events usually calls for unusual means, and are not part of a normal schedule. One can see such highly impacting events as an issue that deals more with crisis management, where recovery scenarios have to be built to return to a normal situation. More frequent but hopefully less severe events in terms of time penalty occur at every instant in a network: leaves or ice on a track causing spinning effects, small delays in operations... Note that the time penalty due to such event is usually low. However, in urban train systems, where departures and arrivals occur at a high rate, accumulation of uncompensated delays happens very fast, and can result in undesirable phenomena (unexpected overtaking, bunching phenomena...).

In this paper, we will only address frequent delays with low individual penalty. Assuming this, delays can be seen as deviations with respect to the expected most probable value, i.e., as *noise*. Note that changes with respect to the expected behavior of a particular train is not a change in the sequence of events occurring along its journey in a network (a train is supposed to follow its journey from its beginning to its end), but rather changes in the occurrence dates of events, or in the way several journeys are interleaved. A standard way to address noise is to define continuous probability distributions for the actual value that a duration (dwell time in stations, trip duration...) may take.

For simplicity, we will consider that random durations appearing in our systems are sampled from closed intervals of the form  $[a, b]$  with  $a < b$  and open intervals of the form  $[a, +\infty)$ . A *probability density function* (PDF) for a continuous random variable  $X$  is a function  $f_X : \mathbb{R} \rightarrow [0, 1]$  that describes the relative likelihood for  $X$  to take a given value. Its integral over the domain of  $X$  is equal to 1. A *cumulative distribution function* (CDF)  $F_X : \mathbb{R} \rightarrow [0, 1]$  for  $X$  describes the probability for  $X$  to take a value less than or equal to a chosen value. We denote by  $\Sigma_{\text{pdf}}$  the set of PDFs,  $\Sigma_{\text{cdf}}$  the set of CDFs, and we only consider PDFs for variables representing durations, i.e., whose domains are included in  $\mathbb{R}_{\geq 0}$ . The CDF of  $X$  can be computed from its PDF as  $F_X(x) = \int_0^x f_X(y) dy$ .

Several standard solutions can be used to model continuous distributions: Gaussian distributions, exponential laws... A drawback of Gaussian distributions is that, considering a duration as a randomly chosen value  $x$  around a pivot value  $a$ , Gaussian distribution attributes the same probability to event  $x < a$  and  $x > a$ . Intuitively, a train has the same probability to be late and to be in advance. Everyday's experience tends to show that the probability for a train to be delayed is higher, so Gaussian distributions does not match exactly the needs for asymmetric distributions in transport systems. In the rest of the paper, we will adopt distributions called truncated polyexponential functions, that have all the features needed to model random delays. A *truncated polyexponential* function is a function  $f(x)$  defined over an interval  $[a, b]$ , such

that  $\int_a^b f(x) dx = 1$ , and is a sum of  $K \in \mathbb{N}$  weighted exponentials of the form:

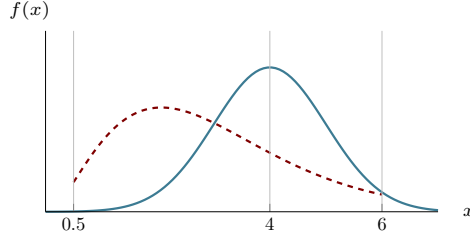
$$f(x) = \begin{cases} \sum_{k \in 1..K} c_k \cdot x^{a_k} \cdot e^{-\lambda_k x} & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$


Figure 2: Expressing probability of a delay with a Gaussian distribution (plain line) and a polyexponential function (dashed line).

Figure 2 illustrates the advantages of using polyexponential functions to model *continuous* distributions. The curves represented in this figure are a standard Gaussian distribution centered around value 4, and a polyexponential function  $f(x) = 0.58 \cdot x^2 \cdot e^{-1.7x} + 0.29 \cdot x^3 \cdot e^{-1.2x}$  defined over domain  $[0.5, 6]$ . As already explained in [3], polyexponential functions can be used to model phase functions, that is functions where distributions have the shape of several bell-like curves centered around several pivot values. This is particularly interesting to represent with a single distribution speed profiles of trains, i.e., very different values corresponding to discrete choice of a targeted travel time perturbed by some noise. This ability allows to model sets of degenerate bells representing quasi-discrete distributions. Last, polyexponential functions compose well and are easy to manipulate: joint distributions are products of exponentials, projections on a variable or on a domain can be done as simple integrations, and remain polyexponential functions.

**Blocking Semantics:** As already mentioned, we consider production systems with limited resources, and in the case of train systems, with strict security requirements. A safety headway requirement imposes that a train cannot enter a track portion if it is occupied by another train. Similar requirements can also be essential for production cells, where a tool usually processes one item at a time. Without care, one can design Petri nets that are not safe a priori (the contents of places is not always 1-bounded). To guarantee such safety requirement, we adapt the semantics of our nets in such a way that a transition cannot fire if its postset contains an already filled place. In an untimed setting, these nets are usually called *elementary nets*. We will however show that in a timed setting, elementary firing rules forces to adapt the notion of urgency.

**Stochastic time Petri nets:** The model that we use to represent timed concurrent systems is a variant of the stochastic Petri nets already proposed by [3], with a *blocking semantics*. Intuitively, a stochastic Petri net is a Petri net with time intervals attached to transitions, and probability density functions associated with these intervals.



**Definition 1 (stochastic time Petri net).** A stochastic time Petri net (STPN for short) is a tuple  $\mathcal{N} = \langle P, T, \bullet(), ()^\bullet, m_0, \text{eft}, \text{lft}, \mathcal{F}, \mathcal{W} \rangle$  where  $P$  is a finite set of places;  $T$  is a finite set of transitions;  $\bullet(): T \rightarrow 2^P$  and  $()^\bullet: T \rightarrow 2^P$  are pre and post conditions depicting from which places transitions consume tokens, and to which places they output produced tokens;  $m_0: P \rightarrow \{0, 1\}$  is the initial marking of the net;  $\text{eft}: T \rightarrow \mathbb{Q}_{\geq 0}$  and  $\text{lft}: T \rightarrow \mathbb{Q}_{\geq 0} \cup \{+\infty\}$  respectively specify the minimum and maximum time-to-fire that can be sampled for each transition; and  $\mathcal{F}: T \rightarrow \Sigma_{\text{pdf}}$  and  $\mathcal{W}: T \rightarrow \mathbb{R}_{> 0}$  respectively associate a PDF and a strictly positive weight to each transition.

For a given place or transition  $x \in P \cup T$ ,  $\bullet x$  will be called the preset of  $x$ , and  $x^\bullet$  the postset of  $x$ . We denote by  $f_t$  the PDF  $\mathcal{F}(t)$ , and by  $F_t$  the associated CDF. To be consistent, we assume that for every  $t \in T$ , the support of  $f_t$  is  $[\text{eft}(t), \text{lft}(t)]$ . The semantics of STPNs can be summarized in a few lines: markings associates a certain number of tokens to places. A transition is enabled if places in its preset are filled. When a transition  $t$  becomes enabled, a value  $x_t$  is sampled from the interval and the distribution attached to  $t$ . Then, this value decreases over time. A transition can fire as soon as its clock has reached 0, its preset is filled, and its postset is empty. We give a precise semantics for STPNs in section 3.

To illustrate how such nets can be used, consider the piece of network in Figure 3. This piece of net represents a part of the train network in Figure 1. Places are used to represent tracks (tokens symbolize trains), and transitions model departures and arrivals of trains. Once at station  $D$  (represented by place  $P_D$ ), a train can be controlled to move towards station  $E$ , it will then enter the track portion from  $D$  to  $E$ , represented by place  $P_{D,E}$ , or it will follow another itinerary, and enter another track (represented by place  $P_{D,\overline{D}}$ ) to move towards station  $\overline{D}$ . This decision is represented by transition  $t_2$ . When a train is on its way to station  $E$ , entering station  $E$  (represented by place  $P_E$ ) is symbolized by firing of transition  $t_3$ . Decision to fire  $t_1$  or  $t_2$  is controlled by additional places (the dotted places  $P_{c1}$  and  $P_{c2}$ ). One can notice that firing  $t_1$  empties  $P_{c1}$  and fills  $P_{c2}$ , and firing  $t_2$  empties  $P_{c2}$  and fills  $P_{c1}$ . With such a simple controller, it is assumed that one train out of two goes from  $D$  to  $E$ , and the other goes from  $D$  to  $\overline{D}$ . The distributions  $f_1, f_2, f_3$  attached to transitions  $t_1, t_2, t_3$  are represented in the right part of Figure 3. Intuitively, a train staying at station  $D$  will leave after a sojourn time comprised between 10 and 30 seconds, as long as its destination place is empty. Sojourn times in place  $P_D$  have different distributions in interval  $[60, 80]$ , depicted by function  $f_1, f_2$ , and that depend on whether the train leaves for station  $E$  or station  $D$ . The trip from  $D$  to  $E$  lasts between 130 and 140 seconds, and the distribution of trip duration is depicted by function  $f_3$ . Now, if a train is ready to leave station  $D$  to go to  $E$  after a sufficient dwell time, but a train already occupies the track portion from  $D$  to  $E$ , then departure is forbidden. In our model, this is implemented by the blocking semantics that says that a transition can fire only when its postset is empty. In particular, in the model of Figure 3, this means that a token will enter place  $P_{D,E}$  (resp. place  $P_{D,\overline{D}}$ ) at earliest 10 time units after  $P_D$  was filled if  $P_{c1}$

(resp.  $P_{c_2}$ ) contains a token. However, even if  $P_{c_1}$  or  $P_{c_2}$  always contain a token, there is no guarantee that the sojourn time of a token in  $P_D$  is smaller than 80 seconds, despite urgency. Indeed, if places  $P_{c_2}$  and  $P_{D,E}$  are filled, transition  $t_1$  has to for  $P_{D,E}$  to be empty to fire, which may occur at earliest 130 seconds and at latest 140 seconds later.

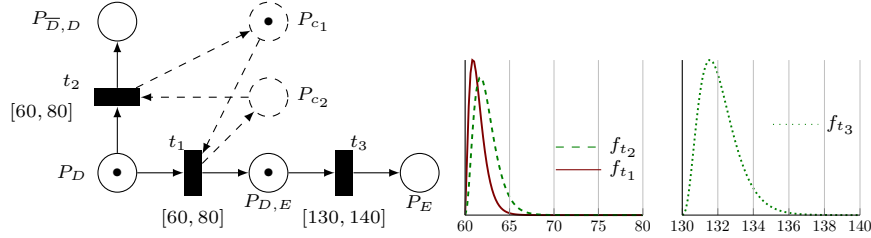


Figure 3: Using STPNs to model trains moves

A possible execution for the STPN of Figure 3 is as follows. Assuming that the value sampled for  $t_1$  is  $x_{t_1} = 62$ , and the value sampled for  $t_3$  is  $x_{t_3} = 132$ , then clock  $x_{t_1}$  expires after 62 time units, but  $t_1$  cannot fire as place  $P_{D,E}$  is filled. Hence, one has to wait 132 time units, fire  $t_3$ , and then fire  $t_1$ . In terms of timed word, this execution can be represented as  $w = (t_3, 132) \cdot (t_1, 132)$ . Equivalently, one can describe this execution as a time process  $TP$  of  $\mathcal{N}$ . Roughly speaking, a time process unfolds the net and associates an execution date to occurrences of transitions (the formal definition of time processes and their construction is given in Section 3). The time process corresponding to timed word  $w$  is given in Figure 4. Note on this example that even if the first occurrences of  $t_1$  and  $t_3$  seem to be concurrent (they occur at the same dates, and are not causally related), the blocking semantics imposes that  $t_3^1$  fires before  $t_1^1$ .

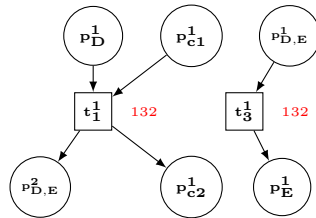


Figure 4: Time process for the net of Figure 3

## 2.2. Schedules, timetables, and their realizability.

Blocking semantics allows to handle safety requirements for systems with critical resources. However, this semantics makes reasoning on systems harder. On our example of Figure 3, a possible wish of designers is that a train stays no longer than 30 seconds at station  $D$ , and one train out of two goes to  $E$ .

315 The second requirement is easily implemented by the simple additional control  
represented by the dashed places and flows. However, nothing guarantees a  
priori the first requirement, i.e., that place  $P_{D,E}$  is emptied at latest 30 seconds  
after a token entered  $P_D$ . So, even if the model seems correct, trains may have  
to wait longer than expected a priori. The objective of this paper is to make  
320 sure that a system can implement a predetermined schedule. The principle of  
imposed schedule appears in many places of our everyday lives. Trains, metros,  
buses, follow predetermined schedules, also called *timetables*. The view that an  
everyday user of transports has of a schedule is only a partial view corresponding  
to the station where she catches a train or a bus, listing all possible departure  
325 hours. Now, this list of dates is simply a projection of more complex objects.

Timetables describe a trajectory for every object: for instance the list of  
stops that a bus has to visit, and the arrival/departure dates from these stops.  
Distinct objects cannot use the same resource at the same date. Hence, although  
individual trajectories can be seen as very linear, and could for instance be  
330 modeled as timed words, dependencies exist: a train can enter a station only  
after its predecessor on the same line has left it.

The natural notion to encode timetable is hence that of partially ordered sets  
of events decorated with dates. Events represent the beginning/end of a task,  
the arrival/departure of a train or bus, etc. Partial ordering among events allows  
335 to account for linearity in individual trajectories and for causal dependencies due  
to exclusive resource use (a train can enter a station only after its predecessor  
has left). The dates decorating these partial order are dates that comply with  
the dependencies, and also with some physical characteristics of the modeled  
systems: a train move from a station  $S_i$  to the following one  $S_{i+1}$  in its itinerary  
340 takes time, which shall be reflected by the dates attached to departures and  
arrivals at different stations.

A schedule can be seen as a solution for a set of constraints over variables  
representing dates of events. Even if a proposed solution satisfies all constraints  
attached to a system, it is usually a high-level view of the overall expected  
345 behavior of a system. This raises the question of whether a particular schedule  
can be effectively realized by the system. As schedules and systems models are  
descriptions of the same system, one can expect the answer to be positive in  
most of cases. However, solution returned by a solver for a constraint problem  
are optimal solutions w.r.t. some criteria, but not necessarily the most plausible  
350 nor the more robust ones. Indeed, for obvious reasons, one cannot ask a bus to  
reach each stop at the earliest possible date: such solution makes systems poorly  
robust to random delays, that necessarily occur in transport systems. Providing  
the ability to check that a schedule is realizable with reasonable chances is hence  
an essential tool to design schedules.

355 In many transport systems, schedules are only ideal behaviors of vehicles, that  
are realized up to some imprecision, and used mainly to guide the system. Systems  
such as bus fleets or metros often deviate from the expected timing prescribed  
by a timetable at peak hours. This deviation mainly concerns occurrence dates  
of events. Re-ordering of events is rare, and mainly occurs in case of a major  
360 failure of the system that forces working in a degraded setting, where following

a timetable makes no sense. Systems are also adaptive: production cells can be equipped with controllers, bus drivers receive instructions to avoid bunching phenomena [12], and metro systems are controlled by regulation algorithms that help trains sticking to predetermined schedules. In this setting, a key question is to assess how much regulation and control is needed to make a system run as expected in its schedule. A first way to answer to this question is to ask the probability that a schedule is realized by the system. In this paper, we give ways to compute the probability that a schedule is realized up to some minor shift in occurrence dates of its events. This measure does not formally include a particular adaptation technique (control, regulation) to help a system stick to a predetermined schedule. However, this measure still makes sense, as it quantifies the need for correction to unregulated systems.

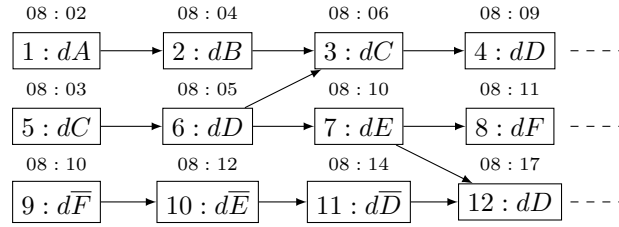


Figure 5: A possible schedule for trains departures in the metro network of Figure 1

Consider the example of Figure 5. This Figure represents the beginning of a schedule for three trains, where only departure dates are planned. The departures for each train are depicted as boxes, carrying a label of the form  $i : dJ$ , where  $i$  is the event number, and  $dJ$  means that this event is a departure from station  $J$ . Furthermore, an execution date is attached to each node. One can see on this drawing that schedules can be seen as partial orders organized along train trajectories. However, there are some dependencies among events from distinct trains: for instance, our schedule imposes that the second departure of the day at station  $D$  (event 6 :  $D$ ) precedes departure 3 :  $C$  from station  $C$ . Similarly, 7 :  $E$  must precede 12 :  $D$ . One can notice that arrivals are not depicted in the schedule, that focuses on departures. Now, a sensible question is: can the network of Figure 1 realize this schedule? More precisely, can the Petri net designed for this network (with additional control places) realize such a schedule starting at 08 : 00 with a train at station  $A$ , a train at station  $C$  and a train at station  $\bar{F}$ ? The answer to this question is not straightforward. Even if the answer is positive, the next step is to ask whether this schedule, with a tolerance of 1 min delay for each departure, is probable enough. If the answer is that the probability to realize our schedule is very low, then this schedule should not be considered as operational.

A schedule describes causal dependencies among tasks, and timing constraints on their respective starting dates. Schedules are defined as decorated partial orders. We allow timing constraints among tasks that are not causally related.

395 **Definition 2 (schedule).** A schedule over a finite alphabet  $\mathcal{A}$  is a quadruple  
 $S = \langle N, \rightarrow, \lambda, C \rangle$  where  $N$  is a set of nodes,  $\rightarrow \subseteq N \times N$  is an acyclic precedence  
relation,  $\lambda : N \rightarrow \mathcal{A}$  is a labeling of nodes, and  $C : N \times N \rightarrow \mathbb{Q}_{>0}$  is a partial  
function that associates a time constraint to pairs of nodes. A dating function  
for a schedule  $S$  is a function  $d : N \rightarrow \mathbb{Q}_{\geq 0}$  that satisfies all constraints of  $C$  and  
400  $\rightarrow : \langle n, n' \rangle \in \rightarrow$  implies  $d(n') \geq d(n)$ , and  $C(n, n') = x$  implies  $d(n') - d(n) \geq x$ .

This model for schedules is inspired from [1, 2]. Intuitively, if  $C(n, n') = x$ ,  
then  $n'$  cannot occur earlier than  $x$  time units after  $n$ , and if  $\langle n, n' \rangle \in \rightarrow$ ,  
then  $n$  (causally) precedes  $n'$ . Constraints model the minimal times needed to  
perform tasks and initiate the next ones in production cells, the times needed  
405 for trains to move from a station to another, etc. A schedule  $S$  is *consistent*  
if the graph  $\langle N, \rightarrow \cup \{ \langle n, n' \rangle \mid C(n, n') \text{ is defined} \} \rangle$  does not contain cycles.  
Obviously, consistent schedules admit at least one dating function. A frequent  
approach is to associate costs to dating functions and to find optimal functions  
that meet a schedule. A cost example is the earliest completion date. Optimizing  
410 this cost amounts to assigning to each node the earliest possible execution date.  
However, these optimal schedules are not the most probable ones. For the earliest  
completion date objective, if an event  $n$  occurs later than prescribed by  $d$ , then  
all its successors will also be delayed. In real systems running in an uncertain  
environment (e.g., with human interactions or influenced by weather conditions),  
415 tight timings are impossible to achieve. Finding a good schedule is hence a  
trade-off between maximization of an objective and of the likelihood to stay  
close to optimal realizations at runtime.

**Realizability:** Accuracy of schedules can be formally defined as follows: We  
want to check whether a consistent schedule  $S$  with its dating function  $d$  can be  
420 realized by a system, described as a stochastic time Petri net  $\mathcal{N}$ . More formally,  
this amounts to verifying that there exists a timed execution  $TP$  of  $\mathcal{N}$  and an  
interpretation function  $\psi$  mapping abstract events of  $S$  onto concrete events of  
 $TP$ , such that causally related events of  $S$  are causally dependent in  $TP$  and  
dates of events in  $TP$  meet the constraints on their abstract representation. In  
425 the rest of the paper, we show how to check realizability of a schedule by a  
STPN, and how to compute a lower bound on the probability that  $S$  is realized  
by  $\mathcal{N}$ . This allows in particular to check that the probability of realization of a  
schedule is not null.

### 3. Semantics of Stochastic Time Petri Nets

430 Roughly speaking, an STPN is a time Petri net with distributions on firing  
times attached to transitions. The semantics of this model describes how tokens  
move from the preset of a transition to its postset. The time that must elapse  
between enabling of a transition and its firing is sampled according to the  
distribution attached to the transition. This model borrows its main features  
435 from [3], with the major difference that the semantics is *blocking*, i.e. it forces  
nets to remain *safe* (1-bounded), as in *elementary nets*. This restriction is  
justified by the nature of the systems we address: in production chains, places

symbolize tools that process only one item at a time. Similarly, when modeling train networks, security requirement impose that two trains cannot occupy the same track portion. Standard time or stochastic Petri nets do not assume a priori bounds on their markings. A way to force boundedness is to add complementary places to the original Petri net and then study it under the usual semantics [13]. However, this trick does not allow to preserve all time and probability issues of the original net. Enforcing a bound via a blocking semantics is hence a practical way to guarantee a priori that models do not allow specification of undesired situations.

Let  $\mathcal{N} = \langle P, T, \bullet(), ()^\bullet, m_0, \text{eft}, \text{lft}, \mathcal{F}, \mathcal{W} \rangle$  be a STPN. A *marking* of  $\mathcal{N}$  is a function that assigns 0 or 1 token to each place  $p \in P$ . We will say that a transition  $t$  is *enabled* by a marking  $m$  iff  $\forall p \in {}^\bullet t, m(p) = 1$ . We denote by  $\text{enab}(m)$  the set of transitions enabled by a marking  $m$ . For a given marking  $m$  and a set of places  $P'$ , we will denote by  $m - P'$  the marking that assigns  $m(p)$  tokens to each place  $p \in P \setminus P'$ , and  $m(p) - 1$  tokens to each place  $p \in P'$ . Similarly, we will denote by  $m + P'$  the marking that assigns  $m(p)$  tokens to each place  $p \in P \setminus P'$ , and  $m(p) + 1$  tokens to each place  $p \in P'$ . Firing a transition  $t$  is done in two steps and consists in: (1) consuming tokens from  ${}^\bullet t$ , leading to a *temporary* marking  $m_{\text{tmp}} = m - {}^\bullet t$ , then (2) producing tokens in  $t^\bullet$ , leading to a marking  $m' = m_{\text{tmp}} + t^\bullet$ .

The blocking semantics can be described by timed and discrete moves among configurations, and can be described informally as follows. A variable  $\tau_t$  is attached to each transition  $t$  of the STPN. As soon as the preset of a transition  $t$  is marked,  $\tau_t$  is set to a random value  $\zeta_t$  (called the *time-to-fire* of  $t$ , or TTF for short) sampled from  $[\text{eft}(t), \text{lft}(t)]$  according to  $f_t$ . We will assume that every CDF  $F_t$  is strictly increasing on  $[\text{eft}(t), \text{lft}(t)]$ , which allows to use *inverse transform sampling* to choose a value (see for instance [14] for details). Intuitively, this TTF represents a duration that *must* elapse before firing  $t$  once  $t$  is enabled. The value of  $\tau_t$  then decreases as time elapses but cannot reach negative values. When the TTF of a transition  $t$  reaches 0, then if  $t^\bullet$  is empty in  $m_{\text{tmp}}$ ,  $t$  becomes *urgent* and has to fire unless another transition with TTF 0 and empty postset fires; otherwise (if  $t^\bullet$  is not empty in  $m_{\text{tmp}}$ ),  $t$  becomes *blocked*: its TTF stops decreasing and keeps value 0, and its firing is delayed until the postset of  $t$  becomes empty; in the meantime,  $t$  can be disabled by the firing of another transition. The semantics of STPNs is *urgent*: time can elapse by durations that do not exceed the minimal remaining TTF of enabled transitions that are not blocked. If more than one transition is urgent, then the transition that fires is randomly chosen according to the respective weights of urgent transitions. We formalize the semantics of STPNs in terms of discrete and timed moves between configurations that memorize markings and TTFs for enabled transitions.

**Definition 3 (configuration of an STPN).** A configuration of an STPN is a pair  $\mathcal{C}_N = \langle m, \tau \rangle$  where  $m$  is a marking, and  $\tau : \text{enab}(m) \rightarrow \mathbb{R}_{\geq 0}$  is a function that assigns a positive real TTF  $\tau_i = \tau(t_i)$  to each transition  $t_i$  enabled by  $m$ . A transition  $t$  is enabled in a configuration  $\langle m, \tau \rangle$  iff it is enabled by  $m$ .

**Definition 4 (firable and blocked transitions).** A transition  $t$  is firable in

485  $\langle m, \tau \rangle$  iff it is enabled by  $m$ , all places of its postset are empty in  $m - \bullet t$ , and its TTF is equal to 0. We denote by  $\text{fira}(\langle m, \tau \rangle)$  the set of firable transitions of  $\langle m, \tau \rangle$ . A transition  $t$  is blocked in  $\langle m, \tau \rangle$  iff it is enabled by  $m$ , its TTF  $\tau(t)$  is equal to 0, and one of its postset places is marked in  $m - \bullet t$ . We denote by  $\text{blk}(\langle m, \tau \rangle)$  the set of blocked transitions in  $\langle m, \tau \rangle$ .

490 **Timed moves:** A timed move  $\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle$  lets a strictly positive duration  $\delta \in \mathbb{R}$  elapse. To be allowed,  $\delta$  must be smaller or equal to all TTFs of transitions enabled by  $m$  and not yet blocked. The new configuration  $\langle m, \tau' \rangle$  decreases TTFs of every enabled and non-blocked transition  $t$  by  $\delta$  time units ( $\tau'(t) = \tau(t) - \delta$ ). Blocked transitions keep a TTF of 0, and  $m$  remains unchanged.

495 **Discrete moves:** A discrete move  $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$  consists in firing a transition  $t$  from a configuration  $\langle m, \tau \rangle$  to reach a configuration  $\langle m' = m - \bullet t + t^\bullet, \tau' \rangle$ . Discrete moves change the marking of a configuration, and sample new times to fire for transitions that become enabled after the move. To define the semantics of discrete moves, we first introduce newly enabled transitions.

500 **Definition 5 (newly enabled transitions).** Let  $m$  be a marking and  $t$  a transition enabled by  $m$ . A transition  $t'$  is newly enabled after firing of  $t$  from  $m$  iff it is enabled by marking  $m' = (m - \bullet t) + t^\bullet$  and either it is not enabled by  $m - \bullet t$  or  $t' = t$ . We denote by  $\text{newl}(m, t) = \text{enab}(m') \cap (\{t\} \cup (T \setminus \text{enab}(m - \bullet t)))$  the set of transitions newly enabled by firing of  $t$  from  $m$ .

505 The transition  $t$  fired during a discrete move is chosen among all firable transitions of  $\langle m, \tau \rangle$ . The new marking reached is  $m' = (m - \bullet t) + t^\bullet$ , and  $\tau'$  is obtained by sampling a new TTF for every newly enabled transition and keeping unchanged TTFs of transitions already enabled by  $m$  and still enabled by  $m'$ .

510 An initial configuration for  $\mathcal{N}$  is a configuration  $\langle m_0, \tau_0 \rangle$  where  $m_0$  is the initial marking of  $\mathcal{N}$ , and  $\tau_0$  attaches a sampled TTF to each transition enabled by  $m_0$ . We will write  $\langle m, \tau \rangle \rightarrow \langle m', \tau' \rangle$  iff there exists a timed or discrete move from  $\langle m, \tau \rangle$  to  $\langle m', \tau' \rangle$ , and  $\langle m, \tau \rangle \xrightarrow{*} \langle m', \tau' \rangle$  iff there exists a sequence of moves leading from  $\langle m, \tau \rangle$  to  $\langle m', \tau' \rangle$ . Interested readers can find detailed operational rules for moves of STPNs in Appendix A.

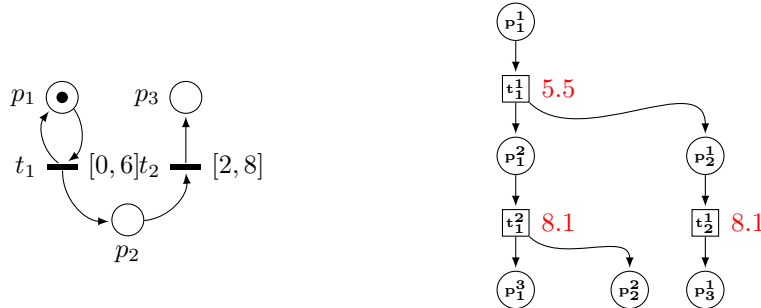


Figure 6: a) An example STPN  $\mathcal{N}_1$  and b) a time process of  $\mathcal{N}_1$

Consider the STPN  $\mathcal{N}_1$  of Figure 6, and suppose that  $\mathcal{N}_1$  is in configuration  $\langle m, \tau \rangle$ , with  $m(p_1) = 1$ ,  $m(p_2) = m(p_3) = 0$ ,  $\tau(t_1) = 5.5$ . From this configuration, one can let 5.5 time units elapse, and then fire  $t_1$ . After this firing, the STPN reaches marking  $m'$  with  $m'(p_1) = m'(p_2) = 1$ ,  $m'(p_3) = 0$ . New TTFs  $d_1, d_2$  are sampled for  $t_1, t_2$ , leading to a configuration  $\langle m', \tau' \rangle$ , where  $\tau'(t_1) = d_1$  and  $\tau'(t_2) = d_2$ . Let us suppose that  $d_1 = 1.5$  and  $d_2 = 2.6$ . Then one can let 1.5 time units elapse, but after this timed move, transition  $t_1$  cannot fire, as place  $p_2$  contains a token.  $\mathcal{N}_1$  is hence in a configuration  $\langle m', \tau'' \rangle$ , where  $\tau''(t_1) = 0$ ,  $\tau''(t_2) = 1.1$ , and  $t_1$  is blocked. After letting 1.1 time units elapse, transition  $t_2$  can fire, leading to marking  $m''(p_1) = m''(p_3) = 1$ ,  $m''(p_2) = 0$ . Once  $t_2$  has fired,  $t_1$  is urgent and firable, and immediately fires, at the same date.

Let us now assign probabilities to STPN moves. Randomness in STPNs semantics mainly comes from sampling of TTFs. However, when several transitions are firable from a configuration, weights are used to determine the probability for a transition to fire first. Timed moves are achieved with probability 1: once TTFs are set, there is a unique configuration allowing discrete moves. In a discrete move  $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$ ,  $m'$  is built deterministically, but  $\tau'$  is obtained by sampling a random value  $\zeta_t$  for each newly enabled transition  $t$ . Each  $\zeta_t$  is chosen according to CDF  $F_t$ , i.e., we have  $\mathbb{P}(\zeta_t \leq x) = F_t(x)$  (for any  $x \in [\text{eft}(t), \text{lft}(t)]$ ). When more than one transition is firable from  $\langle m, \tau \rangle$ , the transition that fires is randomly chosen according to the respective weight of each firable transition: each transition  $t_k$  in  $\text{fira}(\langle m, \tau \rangle)$  has a probability to fire  $\mathbb{P}_{\text{fire}}(t_k) = \mathcal{W}(t_k) / \sum_{t_i \in \text{fira}(\langle m, \tau \rangle)} \mathcal{W}(t_i)$ . Note that, as STPNs have continuous probability laws, the probability to choose a particular value  $\zeta_t$  is the probability of a point in a continuous domain and is hence null. However, in the next sections, we will consider probabilities for events of the form  $\tau(t_i) \leq \tau(t_j)$ , which may have strictly positive probabilities.

STPNs define sequences of moves  $\rho = (\langle m, \tau \rangle \xrightarrow{e_i} \langle m', \tau' \rangle)_{i \in 1 \dots k}$ , where  $e_i$  is a transition name in discrete moves and a real value in timed moves. Leaving probabilities for the moment, STPNs can also be seen as generators for timed words over  $T$ . A *timed word* over an alphabet  $\mathcal{A}$  is a sequence  $\langle a_1, d_1 \rangle \dots \langle a_q, d_q \rangle \dots$  in  $(\mathcal{A} \times \mathbb{R}_{\geq 0})^*$ , where each  $a_i$  is a letter from  $\mathcal{A}$ , each  $d_i$  defines the occurrence date of  $a_i$ , and  $d_1, \dots, d_q$  is an increasing sequence of positive real numbers. Letting  $i_1, \dots, i_q$  denote the indices of discrete moves in  $\rho$ , we can build a timed word  $u_\rho = \langle a_{i_1}, d_1 \rangle \dots \langle a_{i_q}, d_q \rangle \in (T \times \mathbb{R}_{\geq 0})^q$  that associates dates to transitions firings, where  $d_1 = \sum_{j < i_1} e_j$ , and  $d_j = d_{j-1} + \sum_{i_{j-1} < k < i_j} e_k$  for  $j \in \{2, \dots, q\}$ . The *timed language* of an STPN  $\mathcal{N}$  is the set  $\mathcal{L}(\mathcal{N})$  of timed words associated with its sequences of moves. We denote by  $\mathcal{L}_{\leq D}(\mathcal{N})$  the set of words in  $\mathcal{L}(\mathcal{N})$  whose maximal date is lower than  $D$ .

As already highlighted in [15] for TPNs, timed languages give a sequential and interleaved view for executions of inherently concurrent models. A non-interleaved semantics can be defined using *time processes*, i.e., causal nets equipped with dating functions. We recall that *causal nets* are finite acyclic nets of the form  $CN = \langle B, E, \bullet(\cdot), (\cdot)^\bullet \rangle$ , where for every  $b \in B$ ,  $|\bullet b| \leq 1$  and  $|b^\bullet| \leq 1$ . Intuitively, a causal net contains no conflict (pairs of transition with common



places in their presets) nor place receiving tokens from more than one transition.

**Definition 6 (time process).** A time process is a tuple  $TP = \langle CN, \theta \rangle$ , where  
560  $CN = \langle B, E, \bullet(), ()^\bullet \rangle$  is a causal net, and  $\theta : E \rightarrow \mathbb{R}_{\geq 0}$  associates a positive real  
date to transitions of net  $CN$ , and is such that  $\forall e, e' \in E$  with  $e^\bullet \cap {}^\bullet e' \neq \emptyset$  we  
have  $\theta(e) \leq \theta(e')$ . In time processes, places in  $B$  are called conditions, and  
transitions in  $E$  are called events. The depth of a time process is the maximal  
number of events along a path of the graph  $\langle B \cup E, \bullet() \cup ()^\bullet \rangle$ . We will write  $e < e'$   
565 iff  $e^\bullet \cap {}^\bullet e' \neq \emptyset$ , and denote by  $\preceq$  the transitive and reflexive closure of  $<$ .

Intuitively, conditions in  $B$  represent occurrences of places fillings, and events  
in  $E$  are occurrences of transitions firings. We denote by  $\text{tr}(e)$  the transition  
 $t$  attached to an event  $e$ , and by  $\text{pl}(b)$  the place  $p$  associated with a condition  
 $b$ . The flow relations are hence implicit:  ${}^\bullet e = \{b \mid e = \langle X, t \rangle \wedge b \in X\}$ , and  
570 similarly  $e^\bullet = \{b \mid b = \langle p, e \rangle\}$ , and for  $b = \langle p, e \rangle$ ,  ${}^\bullet b = e$  and  $b^\bullet = \{e \in E \mid b \in {}^\bullet e\}$ .  
We will then drop flow relations and simply refer to time processes as triples  
 $TP = \langle B, E, \theta \rangle$ . To differentiate occurrences of transitions firings, an event  
will be defined as a pair  $e = \langle X, t \rangle$ , where  $t$  is the transition whose firing is  
represented  $e$  and  $X$  is the set of conditions it consumes. Similarly, a condition  
575 is defined as a pair  $b = \langle p, e \rangle$ , where  $p$  is the place whose filling is represented by  
 $b$ , and  $e$  is the event whose occurrence created  $b$ .

Given an STPN  $\mathcal{N}$ , for every timed word  $u = \langle a_1, d_1 \rangle \dots \langle a_n, d_n \rangle$  in  $\mathcal{L}(\mathcal{N})$ ,  
we can compute a time process  $TP_u = \langle B, E, \bullet(), ()^\bullet, \theta \rangle$ . The construction  
described below is the same as in [15]. It does not consider probabilities  
580 and, as the construction starts from an executable word, it does not have  
to handle blockings either. We denote by  $TP_u$  obtained from a timed word  
 $u = \langle t_1, d_1 \rangle \langle t_2, d_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$ . It can be built incrementally by adding  
dates transitions one after another. We give a detailed construction in Appendix  
B.

Figure 6-b is an example of a time process for STPN  $\mathcal{N}_1$ . In this ex-  
585 ample, event  $t_i^j$  (resp. condition  $p_i^j$ ) denotes the  $j^{\text{th}}$  occurrence of transi-  
tion  $t_i$  (resp. place  $p_i$ ). This time process corresponds to the time word  
 $u = \langle t_1, 5.5 \rangle \langle t_2, 8.1 \rangle \langle t_1, 8.1 \rangle \in \mathcal{L}(\mathcal{N}_1)$ . It contains causal dependencies among  
transitions (e.g., from  $t_1^1$  to  $t_2^1$ ). Event  $t_1^2$  cannot occur before  $t_2^1$  as  $t_1$  cannot  
590 fire as long as place  $p_2$  is filled. However, this information is not explicit in  
the process. The timed language  $\mathcal{L}(\mathcal{N})$  of a TPN can be reconstructed as the  
set of linearizations of its time processes. In these linearizations, ordering of  
events considers both causality and dates of events:  $e$  must precede  $e' \neq e$  in a  
linearization of a process if  $\theta(e) < \theta(e')$  or if  $e \preceq e'$ . With blocking semantics,  
595 some causality and time-preserving interleavings may not be valid timed words  
of  $\mathcal{L}(\mathcal{N})$ : in the process of Figure 6-b,  $t_1^2$  cannot occur before  $t_2^1$ , even if both  
transitions have the same date. A correct ordering among events with identical  
dates in a process  $TP_u$  can however be found by checking that a chosen ordering  
does not prevent occurrence of other transitions.

#### 600 4. Unfolding of STPNs

A time process emphasizes concurrency but only gives a partial order view of a *single* timed word. Many time processes of  $\mathcal{N}_1$  have the same structure as the process of Figure 6-b, but different dating functions. Indeed, there can be uncountably many time processes with identical structure, but different  
605 real dates. It is hence interesting to consider symbolic (time) processes, that define constraints on events dates instead of exact dates. Similarly, to avoid recomputing the structural part of each symbolic process, we will work with *unfoldings*, i.e., structures that contain all symbolic processes of an STPN, but factorize common prefixes. Symbolic unfoldings were introduced for TPNs in [16]  
610 and used in [17]. In this section, we show how to unfold STPNs with blockings and extract symbolic processes out of this unfolding. Our aim is to find the minimal structure that represents prefixes of all symbolic processes that embed a schedule of known duration. We show that if a system cannot execute arbitrary large sets of events without progressing time, unfolding up to some bounded  
615 depth is sufficient.

**Definition 7 (time progress).** *An STPN  $\mathcal{N}$  guarantees time progress iff there exists  $\delta \in \mathbb{Q}_{>0}$  such that  $\forall t \in T, i \in \mathbb{N}$ , and for every time word  $u = \langle t_1, d_1 \rangle \dots \langle t^i, \theta_1 \rangle \dots \langle t^{i+1}, \theta_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$  where  $t^i$  denotes the  $i^{\text{th}}$  occurrence of  $t$ , we have  $\theta_2 - \theta_1 \geq \delta$ .*

620 Time progress is close to non-Zenoness property, and is easily met (e.g., if no transition has an earliest firing time of 0). The example of Figure 6 does not guarantee time progress as according to the semantics of STPNs, this net allows an arbitrary number of occurrences of transition  $t_1$  to fire at date 0. However, such specification can be considered as ill-formed. Let us consider our motivating  
625 examples: in a production cell, the same kind of processing by a tool necessarily takes time, and conveying an item from a tool to another cannot either be instantaneous. Similarly, in train networks, some time must elapse between two consecutive arrivals of a train at a station, as well as between two consecutive arrivals/departures of the same train. Hence, real-life systems ensure properties  
630 that are often stronger than this time progress property. The example network of Figure 1 will hence necessarily contain exclusively transitions with rational intervals of the form  $[a, b]$  or  $[a, \infty)$ , where  $a > 0$ . An interesting consequence of time progress is that any execution of duration  $\Delta$  of an STPN that guarantees time progress is a sequence of at most  $|T| \cdot \lceil \frac{\Delta}{\delta} \rceil$  transitions. It means that for  
635 most of usual systems that run for a predetermined period (e.g., a metro network operate from 05:00 to 01:00), it is sufficient to consider behaviors of a model up to a certain number of events.

As in processes, unfoldings will contain occurrences of transitions firings (a set of events  $E$ ), and occurrences of places fillings (a set of conditions  $B$ ).  
640 We associate to each event  $e \in E$  positive real valued variables  $\text{doe}(e)$ ,  $\text{dof}(e)$  and  $\theta(e)$  that respectively define the enabling, firability and effective firing date of the occurrence of transition  $\text{tr}(e)$  represented by event  $e$ . Similarly, we associate to each condition  $b$  positive real valued variables  $\text{dob}(b)$  and  $\text{dod}(b)$

that respectively represent the date of birth of the token in place  $\text{pl}(b)$ , and the  
645 date at which the token in place  $\text{pl}(b)$  is consumed. We denote by  $\text{var}(E, B)$  the  
set of variables  $\bigcup_{e \in E} \text{doe}(e) \cup \text{dof}(e) \cup \theta(e) \cup \bigcup_{b \in B} \text{dob}(b) \cup \text{dod}(b)$  (with values in  
 $\mathbb{R}_{\geq 0}$ ). A *constraint* over  $\text{var}(E, B)$  is a boolean combination of atoms of the form  
 $x \bowtie y$ , where  $x \in \text{var}(E, B)$ ,  $\bowtie \in \{<, >, \leq, \geq\}$  and  $y$  is either a variable from  
 $\text{var}(E, B)$  or a constant value. A set of constraints  $C$  over a set of variables  $V$  is  
650 *satisfiable* iff there exists at least one valuation  $v : V \rightarrow \mathbb{R}$  such that replacing  
each occurrence of each variable  $x$  by its valuation  $v(x)$  yields a tautology. We  
denote by  $\text{SOL}(C)$  the set of valuations that satisfy  $C$ .

**Definition 8 (unfolding).** A (structural) unfolding of an STPN  $\mathcal{N}$  is a pair  
 $\mathcal{U} = \langle E, B \rangle$  where  $E$  is a set of events and  $B$  a set of conditions.

655 Unfoldings can be seen as processes with branching. As for processes, each  
event  $e \in E$  is a pair  $e = \langle \bullet e, \text{tr}(e) \rangle$  where  $\bullet e \subseteq B$  is the set of predecessor  
conditions of  $e$  (the conditions needed for  $e$  to occur). A condition  $b \in B$  is  
a pair  $b = \langle \bullet b, \text{pl}(b) \rangle$  where  $\bullet b \subseteq E$  is the predecessor of  $b$ , i.e., the event that  
created condition  $b$ . We assume a dummy event  $\perp$  that represents the origin of  
660 the initial conditions in an unfolding. Function  $\bullet()$ ,  $()^\bullet$ ,  $\text{pl}()$  and  $\text{tr}()$  keep the  
same meaning as for time processes. The main difference between processes  
and unfoldings is that conditions may have several successor events. Using  
relations  $\prec$  and  $\preceq$  as defined for processes, we define the *causal past* of  $e \in E$   
as  $\uparrow e = \{e' \in E \mid e' \preceq e\}$ . A set of events  $E' \subseteq E$  is *causally closed* iff  
665  $\forall e \in E', \uparrow e \subseteq E'$ . This notion extends to conditions. Two events  $e, e'$  are in  
*conflict*, and write  $e \sharp e'$ , iff  $\bullet e \cap \bullet e' \neq \emptyset$ . A set of events  $E' \subseteq E$  is *conflict free* if  
it does not contain conflicting pairs of events. Two events  $e, e'$  are *competing* iff  
 $\text{tr}(e)^\bullet \cap \text{tr}(e')^\bullet \neq \emptyset$  (they fill a common place).

**Definition 9 (pre-processes of an unfolding).** A pre-process of a finite un-  
670 folding  $\mathcal{U} = \langle E, B \rangle$  is a pair  $\langle E', B' \rangle$  such that  $E' \subseteq E$  is a maximal (i.e., there  
is no larger pre-process containing  $E', B'$ ), causally closed and conflict free set  
of events, and  $B' = \bullet E' \cup E'^\bullet$ .  $\mathcal{PE}(\mathcal{U})$  denotes the set of pre-processes of  $\mathcal{U}$ .

Unfolding an STPN up to depth  $K$  is performed inductively, without con-  
sidering time. We will then use this structure to find processes. Timing issues  
675 will be considered through addition of constraints on occurrence dates of events.  
Unfoldings can be built inductively, following the procedure proposed for in-  
stance in [5] and recalled in Appendix C. When building  $\mathcal{U}_0, \dots, \mathcal{U}_K$ , each step  
 $k$  adds new events at depth  $k$  and their postset to the preceding unfolding  
 $\mathcal{U}_{k-1}$ . The construction starts with the initial unfolding  $\mathcal{U}_0 = \langle \emptyset, B_0 \rangle$  where  
680  $B_0 = \{\langle \perp, p \rangle \mid p \in m_0\}$ .

The structural unfolding of an STPN does not consider timing issues nor  
blockings. Hence, an (untimed) pre-process of  $\mathcal{PE}(\mathcal{U}_K)$  needs not be the un-  
timed version of a time process obtained from a word in  $\mathcal{L}(\mathcal{N})$ . Indeed, urgent  
transitions can forbid firing of other conflicting transitions. Similarly, blockings  
685 prevent an event from occurring as long as a condition in its postset is filled.  
They may even prevent events in a pre-process from being executed if a needed

place is never freed. However, time processes of a net  $\mathcal{N}$  can be built from processes of its untimed unfolding. We will show later that, once constrained, time processes of  $\mathcal{N}$  are only prefixes of pre-processes in  $\mathcal{PE}(\mathcal{U}_K)$  with associated  
690 timing function that satisfy requirement on dates that only depend on the considered pre-process. We re-introduce time in unfoldings by attaching constraints to events and conditions of pre-processes. Let  $\mathcal{U}_K = \langle E_K, B_K \rangle$  be the unfolding of an STPN  $\mathcal{N}$  up to depth  $K$ , and let  $E \subseteq E_K$  be a conflict free and causally closed set of events, and  $B = {}^\bullet E \cup E^\bullet$  ( $B$  is contained in  $B_K$ ). We define  $\Phi_{E,B}$   
695 as the set of constraints attached to events and conditions in  $E, B$  (i.e., defined over set of variables  $\text{var}(E, B)$ ), assuming that executions of  $\mathcal{N}$  start at a fixed date  $d_0$ . Constraints in  $\Phi_{E,B}$  set to guarantee:

- (net constraints): occurrence dates of events are compatible with the earliest and latest firing times of transitions in  $\mathcal{N}$ ,
- 700 • (causal precedence): if event  $e$  precedes event  $e'$ , then  $\theta(e) \leq \theta(e')$ .
- (no overlapping conditions): if  $b, b'$  represent occurrences of the same place, then intervals  $[\text{dob}(b), \text{dod}(b)]$  and  $[\text{dob}(b'), \text{dod}(b')]$  have at most one common point,
- (urgency): an urgent transition with empty postset fires if no other urgent  
705 transition fires before. And, in particular, for every event  $e \in E$ , there is no event that becomes fireable and urgent before  $e$ .

Overall,  $\Phi_{E,B}$  is a boolean combination of inequalities. We do not detail its construction here: except for the additional constraint on place occupancy due to blocking semantics, it is almost the solution proposed by [17]. Interested readers  
710 can also find a complete description of the construction of  $\Phi_{E,B}$  in Appendix D. We can now define symbolic processes, and show how instantiation of their variables define time processes of  $\mathcal{N}$ . Roughly speaking, a symbolic process is a prefix of a pre-process of  $\mathcal{U}_K$  (it is hence a causal net) decorated with a *satisfiable* set of constraints on occurrence dates of events. Before formalizing  
715 symbolic processes, let us highlight three important remarks. **Remark 1:** an unfolding up to depth  $K$  misses some constraints on occurrence dates of events due to blockings by conditions that do not belong to  $\mathcal{U}_K$  but would appear in some larger unfolding  $\mathcal{U}_{K'}$ , with  $K' > K$ . We will however show (Prop. 1 and 2) that with time progress assumption, unfolding  $\mathcal{N}$  up to a sufficient depth  
720 guarantees that all constraints regarding events with  $\theta(e) \leq D$  are considered. This allows to define symbolic processes representing the time processes of  $\mathcal{N}$  that are executable in less than  $D$  time units. **Remark 2:** unfoldings consider depth of events, and not their dates. Hence, if a process contains an event  $e$  occurring at some date greater than  $d$ , and another event  $e'$  that belongs to  
725 the same pre-process and becomes urgent before date  $d$ , then  $e'$  must belong to the process, even if it lays at a greater depth than  $e$ . **Remark 3:** Every pre-process  $\langle E, B \rangle$  of  $\mathcal{U}_K$  equipped with constraint  $\Phi_{E,B}$  is not necessarily a symbolic process. Indeed, some events in a pre-process might be competing for the same resource, and make  $\Phi_{E,B}$  unsatisfiable. Consider for instance the

STPN of Figure 7-a). Its unfolding is represented in b), and two of its (symbolic) processes in c) and d). For readability, we have omitted constraints. One can however notice that there exists no symbolic process containing two occurrences of transition  $t_3$ , because conditions  $p_4^1$  and  $p_4^2$  are maximal and represent the same place  $p_4$ .

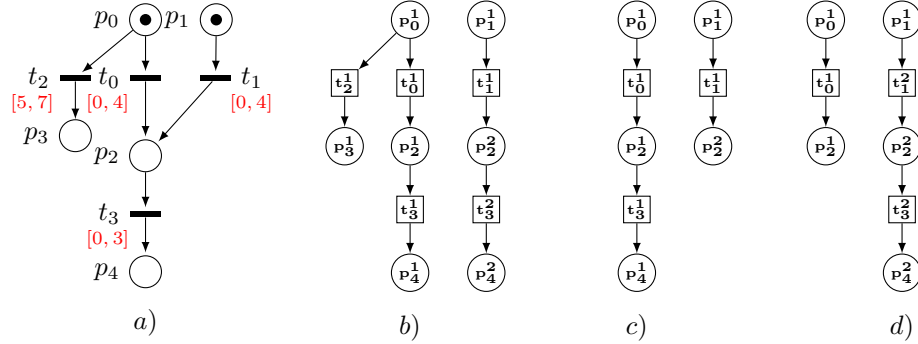


Figure 7: An STPN with conflicts and blockings a), its symbolic unfolding b), and two of its symbolic processes c) and d).

**Definition 10 (prefixes of an unfolding).** Let  $PP = \langle E, B \rangle$  be a pre-process of  $\mathcal{U}_K$ . A symbolic prefix of  $PP$  is a triple  $\langle E', B', \Phi_{E', B'} \rangle$  where  $E' \subseteq E$  is a causally closed set of elements contained in  $E$ , and  $B' = {}^\bullet E' \cup E'^\bullet$ .

Symbolic prefixes are causally closed parts of pre-processes, but their constraints inherited from the unfolding  $\mathcal{U}_K$  may not be satisfiable. Let  $SPP = \langle E', B', \Phi_{E', B'} \rangle$  be a symbolic prefix of pre-process  $PP = \langle E, B \rangle$ . We will say that  $SPP$  is *maximal w.r.t. urgent events firing* iff no more event of  $PP$  **have to** belong to  $SPP$ . This property of  $SPP$  holds if every event  $f \in B'^\bullet \cap E$  that could have become urgent before the last date of all events in  $E'$  was prevented from firing due to blocking. We show in Appendix D that this property of symbolic prefixes can be verified as unsatisfiability of a property  $\Phi_{\max}(f)$  for every  $f \in B'^\bullet \cap E$ .

**Definition 11 (symbolic processes).** A symbolic process of  $\mathcal{U}_K$  is a triple  $\mathcal{E}^s = \langle E', B', \Phi_{E', B'} \rangle$  where  $\langle E', B', \Phi_{E', B'} \rangle$  is a symbolic prefix of some pre-process  $PP = \langle E, B \rangle$  of  $\mathcal{U}_K$ ,  $\Phi_{E', B'}$  is satisfiable, and  $E'$  is maximal w.r.t. urgent events firing in  $PP$ .

A crux in the construction of symbolic processes of  $\mathcal{U}_K$  is to find appropriate maximal and causally closed sets of events with satisfiable constraints. This can be costly: as illustrated by the example of Figure 7, satisfiability of constraints is not monotonous: the constraints for processes in Fig 7-c) and d) are satisfiable. However, adding one occurrence of transition  $t_3$  yields unsatisfiable constraints. Satisfiability of a prefix of size  $n$  hence does not imply satisfiability of a larger prefix of size  $n + 1$ . The converse implication is also false: if a constraint

associated with a prefix of size  $n$  is not satisfiable, appending a new event may introduce blockings that delay urgent transitions, yielding satisfiability of a constraint on a prefix of size  $n + 1$ . So, unsatisfiability of constraints cannot be used as a criterion to stop incremental unfoldings construction.

**Definition 12 (executions of symbolic processes).** Let  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  be a symbolic process of an unfolding  $\mathcal{U}_K$ . An execution of  $\mathcal{E}^s$  is a time process  $TP = \langle E, B, \theta \rangle$  where  $\theta$  is a solution for  $\Phi$ . For a chosen  $\theta$ , we denote by  $\mathcal{E}_\theta^s = \langle E, B, \theta \rangle$  the time process obtained from  $\mathcal{E}^s$ .  $TP = \langle E, B, \theta \rangle$  is a time process of  $\mathcal{U}_K$  if there exists a symbolic process  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  of  $\mathcal{U}_K$  s.t.  $TP$  is an execution of  $\mathcal{E}^s$ .

Informally, symbolic pre-processes select maximal conflict-free sets of events in an unfolding. Symbolic processes extract executable prefixes from symbolic pre-processes, and executions attach dates to events of symbolic processes to obtain time processes. In the rest of the paper, we respectively denote by  $\mathcal{E}^s(\mathcal{U}_K)$  and by  $\mathcal{E}(\mathcal{U}_K)$  the set of symbolic processes and time processes of  $\mathcal{U}_K$ .

We can now show that upon time progress hypothesis, unfoldings and their symbolic processes capture the semantics of STPNs with blockings. Given an STPN that guarantees time progress with a minimal elapsing of  $\delta$  time units between successive occurrences of every transition, and given a maximal date  $D$ , we want to build an unfolding  $\mathcal{U}^D$  of  $\mathcal{N}$  that contains all events that might be executed before  $D$ , but also all places and events which may impact firing dates of these events. We can show that  $\mathcal{U}^D$  is finite and that its processes are of depth at most  $H = \lceil \frac{D-d_0}{\delta} \rceil \cdot |T|$ .

Let  $b = \langle e, p \rangle$  be a condition of an unfolding  $\mathcal{U}_n$  obtained at step  $n$ . Let  $\text{block}(b)$  be the set of conditions that may occur in the same process as  $b$ , represent the same place, and are not predecessors or successors of  $b$  in any unfolding  $\mathcal{U}_{n+k}$  obtained from  $\mathcal{U}_n$ . Clearly, dates of birth and death of conditions in  $\text{block}(b)$  may influence the date of birth and death of  $b$ , or even prevent  $b$  from appearing in the same process as some conditions in  $\text{block}(b)$ . However, in general,  $\text{block}(b)$  need not be finite, and at step  $n$ ,  $\text{block}(b)$  is not fully contained in a pre-process of  $\mathcal{U}_n$ . Fortunately, upon time progress assumption, we can show that elements of  $\text{block}(b)$  that can influence  $\text{dob}(b)$  appear in some bounded unfolding  $\mathcal{U}_K$ .

**Proposition 1.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  time units (between consecutive occurrences of each transition). For every date  $D \in \mathbb{R}_{\geq 0}$  and condition  $b$  in an unfolding  $\mathcal{U}_n$ , there exists  $K \geq n$  s.t.  $\{b' \in \text{block}(b) \mid \text{dob}(b') \leq D\}$  is contained in  $\mathcal{U}_K$ .

This proposition means that if some event cannot occur at  $\text{dof}(e)$  due to a blocking, then one can discover all conditions that prevent this firing from occurring in a bounded extension of the current unfolding.

**Proposition 2.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  time units. The set of time processes executable by  $\mathcal{N}$  in  $D$  time units are prefixes of time processes of  $\mathcal{U}_K$ , with  $K = \lceil \frac{D}{\delta} \rceil \cdot |T|^2$  containing only events with date  $\leq D$ .

## 800 5. Realizability of schedules

We can now address the question of realizability of a high-level description of operations (a schedule  $S$ ) by a system (described by a STPN  $\mathcal{N}$ ). Considered in a purely boolean setting, this question can be rewritten as: is there an execution of  $\mathcal{N}$  that implements  $S$ ? In many cases, a positive answer to this question is not sufficient: as STPNs are equipped with continuous probability distributions, the probability of a particular execution  $TP = \langle E, B, \theta \rangle$  is always 0. A sensible way to address realizability is a quantitative approach requiring that the set of executions of  $\mathcal{N}$  implementing  $S$  has a positive probability. In this section, we first formalize the notion of realization of a schedule by an execution, and define boolean realizability. We then define probabilistic realizability, and sketch how an under-approximation of the probability to realize a schedule can be computed using a transient tree construction [3]. First of all, the connection between high-level description of operations in  $S$  and their implementation in  $\mathcal{N}$  is defined via a realization function.

**Definition 13 (realization function).** *A realization function for a schedule  $S$  and an STPN  $\mathcal{N}$  is a map  $r : \mathcal{A} \rightarrow 2^T$  that associates a subset of transitions from  $T$  to each letter of  $\mathcal{A}$ , and such that  $\forall a \neq a' \in \mathcal{A}, r(a) \cap r(a') = \emptyset$ .*

A realization function describes which low-level actions implement a high-level operation of a schedule. Each letter  $a$  from  $\mathcal{A}$  can be interpreted as an operation performed through the firing of any transition from the subset of transitions  $r(a)$ . Allowing  $r(a)$  to be a subset of  $T$  provides some flexibility in the definition of schedules: in a production cell, for example, a manufacturing step  $a$  for an item can be implemented by different processes on different machines. Similarly, in a train network, a departure of a train from a particular station in the schedule can correspond to several departures using different tracks, or to departure with different speed profiles, which is encoded with several transitions in an STPN. Realization functions hence relate actions in schedules to several transitions in an STPN. The condition  $r(a) \cap r(a') = \emptyset$  prevents ambiguity by enforcing each transition to appear at most once in the image of  $r$ . Note that  $r(\mathcal{A}) \subseteq T$ , that is the realization of a schedule may need many intermediate steps that are depicted in the low-level description of a system, but are not considered in the high-level view provided by a schedule. This allows in particular to define schedules that constrain dates for a subset of events, and leave dates of other events free from any constraint. In the context of a train network, this allows for the verification of realizability of schedules that focus on a subset of stations, e.g., requiring a departure from a choosen station every  $x$  minutes during a normal day of operation. We will call transitions that belong to  $r(\mathcal{A})$  *realizations* of  $\mathcal{A}$ .

**Definition 14 (embedding, realizability).** *Let  $S = \langle N, \rightarrow, \lambda, C \rangle$  be a schedule,  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  be a symbolic process of  $\mathcal{N}$  and  $r : N \rightarrow T$  be a realization function. We say that  $S$  embeds into  $\mathcal{E}^s$  (w.r.t.  $r$  and  $d$ ) and write  $S \hookrightarrow \mathcal{E}^s$  iff there exists an injective function  $\psi : N \rightarrow E$  such that:*

$$\begin{cases}
\forall n \in N, \text{tr}(\psi(n)) \in r(\lambda(n)) & \text{(embedding is consistent with labeling)} \\
\forall \langle n, n' \rangle \in \rightarrow, \psi(n) \preceq \psi(n') & \text{(causal precedence is respected)} \\
\#f \leq \psi(\min(n)), \text{tr}(f) \in r(\mathcal{A}) & \text{(embedding starts on 1st compatible events)} \\
\forall e \leq f \leq g, e = \psi(n) \wedge g = \psi(n'') \wedge \text{tr}(f) \in r(\mathcal{A}) & \text{(embedding "misses" no compatible event)} \\
\Rightarrow \exists n', f = \psi(n') \wedge n \rightarrow^* n' \rightarrow^* n'' &
\end{cases}$$

$S$  embeds in  $\mathcal{E}^s$  iff there is a way to label every node  $n$  of  $S$  by a letter from  $r(\lambda(n))$  and obtain a structure that is contained in some restriction of a prefix of  $\mathcal{E}^s$  to events that are realizations of actions from  $\mathcal{A}$  and to a subset of its causal ordering. This way, a process respects the ordering described in  $S$ , does not “forget” actions, and does not “insert” realizations that are not the image by  $\psi$  of any high-level operation between two mapped realizations, or before the image by  $\psi$  of minimal nodes in the schedule. Note that there can be several ways to embed  $S$  into a process of  $\mathcal{N}$ .

**Definition 15 (realizability).** Let  $d$  be a dating function for a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ ,  $r$  be a realization function. The pair  $(S, d)$  is realizable by  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  (w.r.t.  $r$ ) iff there exists an embedding  $\psi$  from  $S$  to  $\mathcal{E}^s$ , and furthermore,  $\Phi_{\psi, S, d} = \Phi \wedge \bigwedge_{n \in N} \theta(\psi(n)) = d(n)$  is satisfiable.  $(S, d)$  is realizable by  $\mathcal{N}$  (w.r.t.  $r$  and  $d$ ) iff there exists a symbolic process  $\mathcal{E}^s$  such that  $S$  is realizable by  $\mathcal{E}^s$ .

For simplicity, we address realizability of a schedule with respect to a fixed dating function  $d$ . However, realizability of a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$  with constraints  $C$  stands for realizability of any of the dating functions  $d$  meeting constraints  $C$ . Letting  $C_\psi$  denote the conjunction of inequations obtained by replacing every map  $(i, j) \rightarrow v$  in  $C$  by inequation  $\theta(\psi(n_j)) - \theta(\psi(n_i)) \leq v$ , we will say that  $S$  is realizable by  $\mathcal{E}^s$  (w.r.t.  $r$ ) iff there exists an embedding  $\psi$  from  $S$  to  $\mathcal{E}^s$  and  $\Phi_{\psi, S, C} = \Phi \wedge C_\psi$  is satisfiable. We write  $\mathcal{E}^s \models S$  when  $S$  is realizable by  $\mathcal{E}^s$ , and  $\mathcal{N} \models S$  when  $S$  is realizable by  $\mathcal{N}$ . Appendix G gives an algorithm to compute a set  $\Psi_{S, \mathcal{E}^s}$  of embeddings of a schedule  $S$  in a process  $\mathcal{E}^s$ . As soon as an embedding  $\psi \in \Psi_{S, \mathcal{E}^s}$  from  $S$  to a symbolic process  $\mathcal{E}^s$  is obtained, it suffices to prove that  $\Phi_{\psi, S, d}$  (resp.  $\Phi_{\psi, S, C}$ ) is satisfiable to prove that  $S$  is realizable by  $\mathcal{E}^s$ . Realizability hence consists in finding at least one symbolic process of  $\mathcal{N}$  with an appropriate embedding in  $\psi \in \Psi_{S, \mathcal{E}^s}$ . When a maximal occurrence date  $D$  for operations in  $S$  is provided, and when  $\mathcal{N}$  guarantees time progress, such a process is a process of unfolding  $\mathcal{U}_K$  (where  $K$  is the bound given in Prop. 2). We can then compute the set of symbolic processes  $\mathcal{E}^S = \{\mathcal{E}_0^s, \mathcal{E}_1^s, \dots, \mathcal{E}_{N-1}^s\}$  of  $\mathcal{U}_K$  that embed  $S$  and similarly for each  $\mathcal{E}_i^s \in \mathcal{E}^S$ , the set of possible embedding functions  $\Psi_i = \{\psi_{i,0}, \psi_{i,1}, \dots, \psi_{i,N_i-1}\}$  for which constraint  $\Phi_{\psi_{i,j}, S, d}$  is satisfiable.

To illustrate the construction of unfoldings and of processes, let us consider the example of figure 8. This toy example depicts two train carousels: line 1 serves stations  $A, B$  and  $C$ , and line 2 serves stations  $D, B'$  and  $C'$ . Both lines share a common track portion between stations  $B, C$  and  $B', C'$ , and line 1 uses



two trains. The up left picture shows the aspect of both lines and stations, and the bottom left figure a STPN model of this network (we do not show distributions). Stations are represented by places labeled by station names, and track portions between two stations by places labeled by pairs of letters representing the connected stations (e.g., place  $CA$  represents the track from  $C$  to  $A$ ). Transitions consuming tokens from a station place represent trains departures, and transitions consuming tokens from a track place are arrivals. A possible required schedule (middle of the figure) is that one train leaves every 10 time units from station  $A$  on line 1, starting from date 10, and one train leaves station  $B'$  every 10 time units, but starting from date 15. Arrivals of trains and departures from other stations are not represented, and are hence not constrained. Departures from  $A$  are nodes labeled by  $d_A$  and departures from  $B'$  are nodes labeled by  $d_{B'}$ . The rightmost part of the figure is a structural unfolding of the net. We set  $r(d_A) = \{t_5\}$  and  $r(d_{B'}) = \{t_7\}$ . Note that the topmost occurrence of place  $OK$ , that plays the role of a boolean flag in a critical section can be both consumed by occurrences  $t_1^1$  and  $t_1^2$  of transition  $t_1$ , which is a standard conflict. Note also that events  $t_4^1$  and  $t_4^2$  output a token in place  $A$ . Even if these events are not in conflict, due to non-blocking semantics, their firing dates may influence one another. The way operations of the schedule inject in a process of the net is symbolized by dotted lines. Notice that if  $t_1^5$  has to fire at date 10, then according to intervals attached to transitions,  $t_4^1$  has to fire at date 5. This means that there exists a unique way to guarantee a departure from station  $A$  at date 10, which is to sample the smallest trip durations from  $C$  to  $A$  and the smallest possible dwell time at station  $A$ . The probability of such kind of schedule with precise dates is obviously 0.

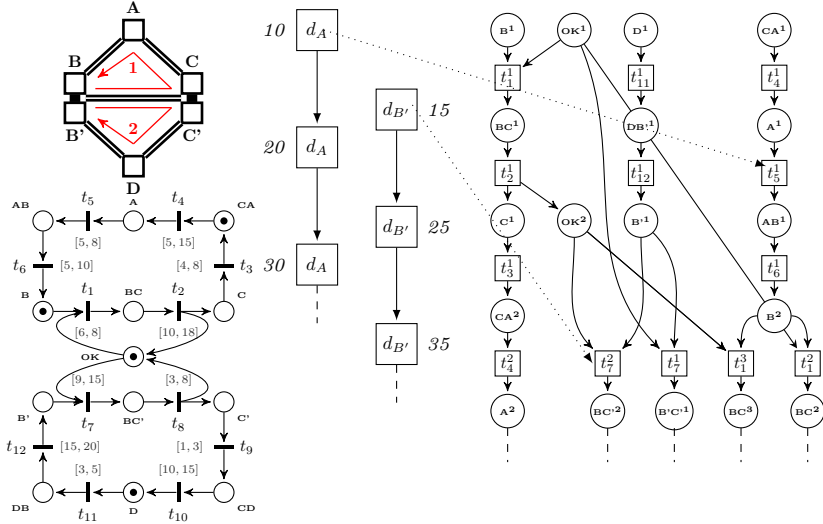


Figure 8: Realizability of a schedule for a metro network with two lines and a shared track.

The example of Figure 8 shows that boolean realizability characterizes an

embedding that is consistent with time constraints, but not the probability to  
905 realize a schedule. Consider the STPN of Figure 9. This net has two symbolic  
processes:  $\mathcal{E}_1^s$  in which transition  $t_1$  fires, and  $\mathcal{E}_2^s$  in which  $t_2$  fires. The probability  
of process  $\mathcal{E}_1^s$  is the probability that a value  $v_1$  sampled to assign a TTF for  $t_1$  is  
smaller or equal to another value  $v_2$  sampled independently to assign a TTF for  
 $t_2$ . Clearly, the probability that  $v_1 \leq v_2$  is equal to the probability that  $v_1 \in [0, 1]$   
910 (and is hence equal to 1). The probability of the second process  $\mathcal{E}_2^s$  is equal to  
the probability that  $v_1 \geq v_2$ , but the set of values allowing this inequality is  
restricted to a single point  $v_1 = 1, v_2 = 1$ . Conforming to continuous probability  
distributions semantics, the probability of this point, and consequently the  
probability of executing a time process that is consistent with constraints in  $\mathcal{E}_2^s$  is  
915 0. A schedule  $S$  composed of a single node  $n$  with a realization function such that  
 $r(\lambda(n)) = \{t_2\}$  and a date  $d(n) = 1$  are realizable according to Definition 14, but  
with *null probability*. This is not a surprise: requiring a schedule to be realized  
with an exact timing in a continuous probability setting leads to realizations with  
null probabilities. Let us slightly change the example of Figure 9-a). We now  
920 assign interval  $[0, 3]$  to transition  $t_1$  in the STPN and interval  $[1, 4]$  to transition  
 $t_2$ . We keep the same schedule  $S$  and realization function  $r$ , but require that  
 $d(n) = 2$ . The probability that  $t_2$  fires from the initial marking is equal to the  
probability that  $v_1 \geq v_2$ , which is not null (we explain in Appendix I how to  
compute the probability of such domain and the joint probability of  $v_1, v_2$ ), and  
925 is equal to the joint probability of values of  $v_1, v_2$  laying in domain  $v_1 \geq v_2$   
depicted by the grey zone in Figure 9-b). However, within this continuous  
domain of possible values, the probability to fire  $t_2$  exactly at precise date 2 as  
required by dating function  $d$  is still null. Nevertheless, if  $t_2$  is allowed to fire  
at date 2 with some imprecision  $\alpha$ , then the probability to realize the expected  
930 schedule is equal to the integration of the joint probability distribution over the  
domain where  $(v_1 \geq v_2) \wedge (2 - \alpha \leq v_2 \leq 2 + \alpha)$  (represented as a dashed part in  
Figure 9-b), which can be strictly positive if distributions attached to  $t_1$  and  
 $t_2$  are properly set. Note that requiring dates to be implemented up to some  
imprecision does not necessarily increase the probability to realize a schedule: in  
935 the example of Figure 9-a with intervals  $[0, 1]$  and  $[1, 2]$  the only way to realize  
the schedule  $S$  mentioned before with date  $d(n) = 1$  is to execute the unique  
time process in which  $t_2$  fires at date 1, and the probability of this process is  
null. More generally, the probability to realize a schedule when the embedding  
relation leaves a single possible occurrence date for at least one event in the  
940 chosen symbolic process is always null.

Boolean realizability is a first step to check that a schedule and an implemen-  
tation are not totally orthogonal visions of a system. However, examples 8 and 9  
demonstrate that it is not precise enough. They also show that boolean realizabil-  
ity up to imprecision still allows to consider sets of processes with null probabilities  
945 as realizations of a schedule. An accurate notion of realizability should require  
that schedules embed into symbolic processes of  $\mathcal{U}_K$  with strictly positive proba-  
bility **and** up to some admissible imprecision on dates of events, bounded by  
some value  $\alpha \in \mathbb{Q}_{\geq 0}$ . An operation  $x$  in a schedule should now be implemented

by a occurrence  $t_i^j$  of a transition  $t$  at date  $\theta(t_i^j) \in [\max(d(x) - \alpha, 0), d(x) + \alpha]$ .  
 950 Once an injection  $\psi$  from a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$  to a symbolic process  $\mathcal{E}^s$  is found, the constraint to obtain realizability of a dating function  $d$  up to imprecision of  $\alpha$  becomes:  $\Phi_{\psi, S, d \pm \alpha} = \Phi \wedge \bigwedge_{n \in N} \max(d(n) - \alpha, 0) \leq \theta(\psi(n)) \leq d(n) + \alpha$ . One can similarly require constraints  $C$  in  $S$  to be realized up to imprecision of  $\alpha$ , that is, require that constraints of the form  $d(n_j) - d(n_i) \leq v$  imposed by map  $C$   
 955 are implemented in the low level net by a process satisfying a relaxed constraint of the form  $\theta(\psi(n_j)) - \theta(\psi(n_i)) \leq v + \alpha$ . This notion of realization up to bounded imprecision is more natural than boolean realizability. Indeed, for systems such as train networks, one cannot expect a schedule to be precisely realized (trains are subject to random delays), but rather that differences between realized and  
 960 scheduled dates are usually not too important.

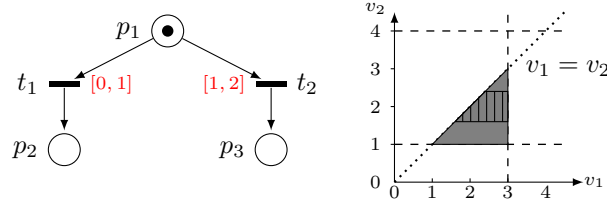


Figure 9: a) An example STPN b) A domain for  $\tau(t_1), \tau(t_2)$  allowing firing of  $t_2$ , assuming  $I(t_1) = [0, 3]$  and  $I(t_2) = [1, 4]$ .

**Definition 16 (probabilistic realizability).** Let  $d$  be a dating function with maximal date  $D$  for a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ ,  $r$  be a realization function. The pair  $(S, d)$  is realizable with non-null probability (w.r.t.  $r$ ) up to imprecision  $\alpha$  iff there exists an embedding  $\psi$  of  $S$  into a symbolic process  $\mathcal{E}^s$  of  $\mathcal{U}_K$  such  
 965 that:  $\mathbb{P}(\mathcal{E}^s \wedge \text{SOL}(\Phi_{\psi, S, d \pm \alpha})) > 0$ .

Intuitively, this definition requires that a symbolic process embeds  $S$ , and that the probability that this process is executed and satisfies all timing constraints imposed by the STPN and by the dating function is strictly positive. This probability can be evaluated using a transient execution tree, as proposed in [3].  
 970 Roughly speaking, nodes of this tree are abstract representations of time domains for sampled values attached to enabled transitions (this is the usual notion of *state class*, already used in [18, 19] to analyze time Petri nets). In addition to state classes, transient tree nodes contain abstract representations of continuous probability distributions over the time domains defined by these classes. The  
 975 probability to fire a particular transition from a state and move to a successor node is computed as an integration over the time domain allowing this transition to fire first. On the example of Figure 9 (with intervals  $[0, 3]$  and  $[1, 4]$ , this corresponds to integration of a joint distribution for values  $v_1, v_2$  over the domain in grey. The distributions attached to transitions of STPNs are polyexponential  
 980 functions. Beyond their expressive power, polyexponential functions are closed under projections, integrations, or multiplication. Further, (joint) distributions of clock values in a node can always be encoded as polyexponential functions.

This way, one can iteratively build a tree whose nodes contain state classes and  
 distributions over these classes. The parent-child relation is driven by which  
 985 transition  $t$  fires from the parent node (this implies projecting existing values on  
 a part of the class of the parent that allows firing  $t$  first), and which transitions  
 are newly enabled after firing  $t$  (yielding re-sampling of TTFs for the newly  
 enabled transitions). As time progress is guaranteed in our model, a *finite*  
 tree representing executions of an STPN or of one of its processes up to some  
 990 bounded duration can always be built. As shown in [3], after this transient  
 tree construction the (sum of) probabilities attached to paths of the tree can  
 be used to compute the probability of properties such as safety of a system  
 within a bounded horizon. In our case, the sum of probabilities of all paths  
 that end with the execution of a chosen symbolic process gives the probability  
 995 to realize this process. If this probability is not null, then there is a positive  
 probability to realize the considered schedule. Note that the computed value  
 is only a lower bound for the exact probability to realize a schedule: indeed  
 there can be more that one process realizing a schedule. However, computing  
 the exact probability is rather involved, as distinct realizations of a schedule are  
 1000 not necessarily independent. Details on the construction of this transient tree  
 are provided in Appendix H.

## 6. Conclusion

**Related work:** we have addressed realizability of partially ordered timed  
 schedules by timed and stochastic concurrent systems with safety requirements.  
 1005 Realizability in a timed setting has formerly been addressed as a timed game  
 problem [20], with a boolean answer. The objective in this work is to check  
 whether a player in a timed game has a strategy to ensure satisfaction of a  
 formula written in a timed logic called Metric Interval Temporal Logic. This  
 work could be used to answer a boolean realization question, by translating a  
 1010 schedule to a formula. However, this work of [20] lies in an interleaved setting:  
 a sequential formula cannot differentiate interleaved and concurrent actions. It  
 does not address randomness in systems and hence cannot quantify realizability.  
 Scheduling of train networks was already addressed as a constraint satisfaction  
 problem [1]. The input of the problem is given as an alternative graph (that can  
 1015 be seen as some kind of unfolding of a systems's behavior, decorated with time  
 constraints). The solutions proposed in [1] use a branch and bound algorithm to  
 return an optimal schedule for the next 2 hours of operation of a train network,  
 but do not consider randomness. Note however that this algorithm is efficient  
 enough to be used online to guide decisions of a scheduling system. Stability of  
 1020 timetables in an environment with random perturbations was considered in [21]:  
 a schedule is considered as an immutable ordering on trains, and delays are  
 modeled as probability distributions. Reliability of a given timetable (with fixed  
 dates) is then defined via probability measures (probability that a given number  
 of trains gets late by more than a fixed threshold...).

1025 Realizability is also close to *diagnosis*. Given a log (a partial observation  
 of a run of a system), and a model for this system, diagnosis aims at finding

all possible runs of the model of the system whose partial observation complies with the log. Considering a log as a schedule, the ability to compute a diagnosis implies realizability of this high-level log by the model. Diagnosis was addressed for stochastic Petri nets in [22]. In this work, the likelihood of a process that complies with an observation is evaluated, and time is seen as a sequence of discrete instants. Diagnosis was addressed for parameterized Petri nets in [6]. The proposed solution unfolds a parameterized Petri net to find explanations for an observed log. Time Petri nets can be considered as a particular case of this parameterized model. [23] proposes temporal patterns called chronicles that represent possible evolutions of an observed system. A chronicle is a set of events, linked together by time constraints. The diagnosis framework explains stream of time-stamped events as combinations of chronicles. Assembling chronicles is some kind of timed unfolding. However, event streams are not a concurrent model, and chronicles extraction does not consider randomness.

Schedulability can also be seen as conformance of an expected behavior (the schedule) to an implementation (the STPN model). Conformance was defined as a timed input/output conformance relation (tIOCO) relation between timed input/output automata in [24]. More precisely, timed automaton  $\mathcal{A}_1$  is in tIOCO relation with timed automaton  $\mathcal{A}_2$  iff after some timed word, the set of outputs produced by  $\mathcal{A}_1$  is included in the outputs produced by  $\mathcal{A}_2$ . This relation cannot be verified in general (as inclusion of timed automata languages is not decidable), but can be tested. Boolean realizability can be seen as some kind of conformance test. Note however that tIOCO is defined for an interleaved timed model without probabilities.

**Assessment:** The techniques described in this work first build an unfolding  $\mathcal{U}_K$  up to a depth  $K$  that depends on the maximal date appearing in the schedule, find symbolic processes of  $\mathcal{U}_K$  that embed the schedule, and then check that at least one of them has a strictly positive probability to be executed. In a purely boolean setting, this amounts to answering a satisfiability question for a combination of linear inequations, and in a probabilistic setting, this means computing a transient execution tree to obtain a quantitative answer. A first question is whether all these steps can be factorized. A second question is of course complexity of this algorithm.

A noticeable fact is that satisfiability of constraints associated with processes and embeddings and unfolding are rather distinct parts of the proposed method. Usually, unfoldings can be stopped as soon as some criterion is met (completeness for construction of finite prefixes[4, 5], satisfaction of a property. . . As shown in Section 4, due to blocking semantics, satisfiability of constraints is not monotonous w.r.t. the size of unfoldings, and hence cannot be used as a criterion to stop unfolding. However, embedding verification and unfolding can be done jointly: one can stop a branch of unfolding as soon as a schedule does not embed in the pre-process on this branch.

Regarding complexity, the size of unfoldings and hence of constraints associated with symbolic processes can grow exponentially with depth. However, these constraints can be simplified to refer only to event variables. One can also notice that atoms in constraints are rather simple inequalities. Hence,

satisfiability of constraints could remain simple linear problems in most cases. However, due to blocking semantics, constraints also contain disjunctions. This  
 1075 may raise combinatorial issues and make satisfiability of constraint costly. The cost of the algorithm to compute realization probability for processes is also an issue. We use the transient tree construction of [3], that builds a symbolic but *interleaved* representation of some process. This is obviously very costly. We are currently investigating ways to evaluate probabilities of symbolic processes  
 1080 in a non-interleaved setting. Note also that the algorithm designed so far only computes lower bounds for realization probability.

As future work, we would like to implement and improve this realizability verification framework, and use it as a basis to prove more properties. For instance, it is interesting to prove that a schedule can be realized while ensuring  
 1085 that the overall sum of delays w.r.t. the expected schedule does not exceed some threshold. Another improvement would be to provide means to compute an exact value for the realization probability, and not only lower bounds.

## References

- 1090 [1] A. D’Ariano, D. Pacciarelli, M. Pranzo, A branch and bound algorithm for scheduling trains in a railway network, *European Journal of Operational Research* 183 (2007) 643–657.
- [2] A. D’Ariano, M. Pranzo, I. Hansen, Conflict resolution and train speed coordination for solving real-time timetable perturbations, *IEEE Trans. on Intelligent Transportation Systems* 8 (2007) 208–222.
- 1095 [3] A. Horváth, M. Paolieri, L. Ridi, E. Vicario, Transient analysis of non-Markovian models using stochastic state classes, *Performance Evaluation* 69 (2012) 315–335.
- [4] K. L. McMillan, A technique of state space search based on unfolding, *Formal Methods in System Design* 6 (1995) 45–65.
- 1100 [5] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan’s unfolding algorithm, *Formal Methods in System Design* 20 (2002) 285–310.
- [6] T. Chatain, C. Jard, Symbolic diagnosis of partially observable concurrent systems, in: *FORTE’04*, volume 3235 of *LNCS*, pp. 326–342.
- 1105 [7] T. Chatain, C. Jard, Complete finite prefixes of symbolic unfoldings of safe time Petri nets, in: *ICATPN’06*, 2006, pp. 125–145.
- [8] P. Merlin, A Study of the Recoverability of Computing Systems, Ph.D. thesis, University of California, Irvine, CA, USA, 1974.
- [9] R. R. Razouk, C. V. Phelps, Performance analysis using timed petri nets, in: *Protocol Specification, Testing and Verification IV*, pp. 561–576.

- 1110 [10] M. Boyer, M. Diaz, Multiple enabledness of transitions in Petri nets with time, in: Proc. of PNPM 2001, IEEE, 2001, pp. 219–228.
- [11] S. Akshay, B. Genest, L. Hélouët, Decidable classes of unbounded Petri nets with time and urgency, in: Proc. of PETRI NETS 2016, volume 9698 of *LNCS*, pp. 301–322.
- 1115 [12] D. Bartholdi, J.J. ans Eisenstein, A self-coordinating bus route to resist bus bunching, *Transportation Research, Part B* 46 (2012) 481–491.
- [13] J. Cortadella, M. Kishinevsky, L. Lavagno, A. Yakovlev, Synthesizing Petri nets from state-based models, in: ICCAD’95, pp. 164–171.
- [14] R. Y. Rubinstein, D. Kroese, Simulation and the Monte Carlo Method, 1120 Wiley, 2 edition, 2008.
- [15] T. Aura, J. Lilius, Time processes for time Petri nets, in: ICATPN’97, volume 1248 of *LNCS*, pp. 136–155.
- [16] A. Semenov, A. Yakovlev, Verification of asynchronous circuits using time Petri net unfolding, in: DAC, pp. 59–62.
- 1125 [17] T. Chatain, C. Jard, Time supervision of concurrent systems using symbolic unfoldings of time Petri nets, in: FORMATS’05, volume 3829 of *LNCS*, pp. 196–210.
- [18] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time Petri nets, *IEEE Trans. Software Eng.* 17 (1991) 259–1130 273.
- [19] D. Lime, O. Roux, Model checking of time Petri nets using the state class timed automaton, *Discrete Event Dynamic Systems* 16 (2006) 179–205.
- [20] L. Doyen, G. Geeraerts, J. Raskin, J. Reichert, Realizability of real-time logics, in: FORMATS’09, volume 5813 of *LNCS*, pp. 133–148.
- 1135 [21] M. Carey, Ex ante heuristic measures of schedule reliability, *Transportation Research* 33 (1999) 473–494.
- [22] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard, Fault detection and diagnosis in distributed systems: An approach by partially stochastic Petri nets, *Discrete Event Dynamic Systems* 8 (1998) 203–231.
- 1140 [23] C. Dousson, Extending and unifying chronicle representation with event counters, in: ECAI’02, pp. 257–261.
- [24] M. Krichen, S. Tripakis, Conformance testing for real-time systems, *Formal Methods in System Design* 34 (2009) 238–304.

## Appendix A. Formal semantics of STPNs

1145 Timed moves are formally defined by the following rule:

$$\frac{\begin{array}{l} \delta > 0 \\ \wedge \quad \forall t \in \text{enab}(m) \setminus \text{blk}(\langle m, \tau \rangle), \tau(t) \geq \delta \wedge \tau'(t) = \tau(t) - \delta \\ \wedge \quad \forall t \in \text{blk}(\langle m, \tau \rangle), \tau'(t) = \tau(t) \end{array}}{\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle}$$

A transition  $t'$  is *persistent* after firing of  $t$  from  $m$  iff it is enabled in  $m$ ,  $t \neq t'$ , and  $t'$  is enabled in  $m - \bullet t$ . We denote by  $\text{pers}(m, t) = (\text{enab}(m) \cap \text{enab}(m_{tmp})) \setminus \{t\}$  the set of persistent transitions after firing of  $t$  from  $m$ .

Discrete moves are formally defined by the following operational rule:

$$\frac{\begin{array}{l} t \in \text{fira}(\langle m, \tau \rangle) \\ \wedge \quad m' = (m - \bullet t) + t^\bullet \\ \wedge \quad \forall t_i \in \text{pers}(m, t), \tau'(t_i) = \tau(t_i) \\ \wedge \quad \forall t_i \in \text{newl}(m, t), \tau'(t_i) \in [\text{eft}(t), \text{lft}(t)] \end{array}}{\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle}$$

## 1150 Appendix B. Construction of time processes

The time process  $TP_u$  obtained from a timed word  $u = \langle t_1, d_1 \rangle \langle t_2, d_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$  is built inductively as follows. We assume a dummy initial event  $\perp$  that initializes the initial contents of places according to  $m_0$ . We start from the initial process  $TP_0 = \langle B_0, E_0, \theta_0 \rangle$  with a set of conditions  $B_0 = \{(p, \perp) \mid p \in m_0\}$ , a set of events  $E_0 = \{\perp\}$ , and a function  $\theta_0 : \{\perp\} \rightarrow \{0\}$ .

1155 Let  $TP_{u,i} = \langle B_i, E_i, \theta_i \rangle$  be the time process built after  $i$  steps for the prefix  $\langle t_1, d_1 \rangle \dots \langle t_i, d_i \rangle$  of  $u$ , and let  $\langle t, d_{i+1} \rangle$  be the  $(i+1)^{th}$  entry of  $u$ . We denote by  $\text{last}(p, E_i, B_i)$  the last occurrence of place  $p$  in  $TP_{u,i}$ , i.e., the only condition  $b = \langle p, e \rangle$  with an empty postset. Then, we have  $E_{i+1} = E_i \cup \{e\}$ , where  $e = \langle t, X \rangle$  with  $X = \{b \mid b = \text{last}(p, E_i, B_i) \wedge p \in \bullet t\}$  and  $B_{i+1} = B_i \cup \{\langle p, e \rangle \mid p \in t^\bullet\}$ . We also set  $\theta(e) = d_{i+1}$ . The construction ends with  $TP_u = TP_{u,|u|}$ .

## Appendix C. Structural unfolding of a STPN

We say that a condition  $b \in B$  is *maximal* in  $\mathcal{U} = \langle E, B \rangle$  or in a pre-process of  $\mathcal{U}$  when it has no successor event ( $b^\bullet = \emptyset$ ), and denote the set of maximal conditions of  $B$  by  $\text{max}(B)$ . As for time processes construction, given a finite pre-process  $\langle E', B' \rangle \in \mathcal{PE}(\mathcal{U})$ , and a place  $p$  of the considered STPN, we denote by  $\text{last}(p, E', B')$  the maximal occurrences of place  $p$  w.r.t.  $\prec$  in  $\langle E', B' \rangle$ . Pre-processes of an unfolding are conflict free sets of events and conditions. They represent potential processes (executions) of  $\mathcal{N}$  when timing constraints are forgotten. A *cut* of a pre-process is an unordered set of conditions. As this set of conditions originates from a pre-process, conditions in a cut have no conflicting events in their causal past. They represent place contents that can be consumed



by the next firable transitions at some point in an execution. We denote by  $Cuts(E, B)$  the set of cuts of pre-process  $\langle E, B \rangle$ . Unfolding of a Petri net simply  
 1175 consists in successively appending transitions to already built processes, i.e. to cuts of these proccesses.

**Structural unfolding:** Following [5], we inductively build unfoldings  $\mathcal{U}_0, \dots, \mathcal{U}_K$ . Each step  $k$  adds new events at depth  $k$  and their postset to the preceding unfolding  $\mathcal{U}_{k-1}$ . We start with the initial unfolding  $\mathcal{U}_0 = \langle \emptyset, B_0 \rangle$  where  
 1180  $B_0 = \{ \langle \perp, p \rangle \mid p \in m_0 \}$ . Each induction step that builds  $\mathcal{U}_{k+1}$  from  $\mathcal{U}_k$  adds new events and conditions to  $\mathcal{U}_k$  as follows. Letting  $\mathcal{U}_k = \langle E_k, B_k \rangle$  be the unfolding obtained at step  $k$ , we have  $\mathcal{U}_{k+1} = \langle E_k \cup \hat{E}, B_k \cup \hat{B} \rangle$  where  $\hat{E} \triangleq \{ \langle B, t \rangle \in (2^{B_k} \times T) \setminus E_k \mid \exists \langle X, Y \rangle \in \mathcal{PE}(\mathcal{U}_k), B \subseteq Cuts(X, Y), \bullet t = pl(B) \}$ , and  $\hat{B} \triangleq \{ \langle e, p \rangle \in \hat{E} \times T \mid e = \langle B, t \rangle \in \hat{E} \wedge p \in t^\bullet \}$ . Intuitively,  $\hat{E}$  adds an occurrence  
 1185 of a transition if its preset is contained in the set of conditions representing the last occurrences of places contained in some pre-process of  $\mathcal{U}_k$ , and  $\hat{B}$  adds the conditions produced by  $\hat{E}$ .

#### Appendix D. Constraints to reintroduce time in processes

Let  $\mathcal{U}_K = \langle E_K, B_K \rangle$  be the unfolding of an STPN  $\mathcal{N}$  up to depth  $K$ , and let  
 1190  $E \subseteq E_K$  be a conflict free and causally closed set of events, and  $B = \bullet E \cup E^\bullet$  ( $B$  is contained in  $B_K$ ). We define  $\Phi_{E,B}$  as the set of constraints attached to events and conditions in  $E, B$ , assuming that executions of  $\mathcal{N}$  start at a fixed date  $d_0$ . Constraints must be set to guarantee that occurrence dates of events are compatible with the earliest and latest firing times of transitions  
 1195 in  $\mathcal{N}$ , and that urgency or blocking is never violated. Let us first define the constraints associated with each condition  $b = \langle e, p \rangle$ . Recalling that variable  $\text{dob}(b)$  represents the date at which condition  $b$  is created,  $\Phi_{E,B}$  must impose that for every  $b \in B_0$ ,  $\text{dob}(b) = d_0$ .

For all other conditions  $b = \langle e, p \rangle$ , as the date of birth is exactly the occurrence  
 1200 date of  $e$ , we set  $\text{dob}(b) = \theta(e)$  for every  $b = \langle e, p \rangle$ . Despite this equality, we will use both variables  $\theta(e)$  and  $\text{dob}(b)$  for readability reasons. Recall that  $\text{dod}(b)$  is a variable that designates the date at which a place is emptied by some transition firing,  $\text{dod}(b)$  is hence the occurrence date of an event that has  $b$  as predecessor. Within a conflict free set of events, this event is unique. In the  
 1205 considered subset of conditions  $B$ , several conditions may represent fillings of the same place, and  $B$  can hence be partitioned into  $B_1 \uplus B_2 \uplus \dots \uplus B_{|P|}$ , where conditions in  $B_i$  represent fillings of place  $p_i$ . Due to blocking semantics, all conditions in a particular subset  $B_i = \{b_{i,1}, b_{i,2}, \dots, b_{i,k}\}$  must have disjoint existence dates, that is for every  $j, j' \in \{1, 2, \dots, k\}$  with  $j \neq j'$ , the intersection between  $[\text{dob}(b_{i,j}), \text{dod}(b_{i,j})]$  and  $[\text{dob}(b_{i,j'}), \text{dod}(b_{i,j'})]$  is either empty, or limited  
 1210 to a single value. This constraint can be encoded by the disjunction:

$$\begin{aligned} \text{no-overlap}(b_{i,j}, b_{i,j'}) = & \\ \left\{ \begin{array}{ll} \text{dob}(b_{i,j}) \leq \text{dob}(b_{i,j'}) \vee \text{dod}(b_{i,j'}) \leq \text{dob}(b_{i,j}) & \text{if } b_{i,j}^\bullet \neq \emptyset \wedge b_{i,j'}^\bullet \neq \emptyset, \\ \text{dob}(b_{i,j}) \leq \text{dob}(b_{i,j'}) & \text{if } b_{i,j}^\bullet \neq \emptyset \wedge b_{i,j'}^\bullet = \emptyset, \\ \text{dod}(b_{i,j'}) \leq \text{dob}(b_{i,j}) & \text{otherwise.} \end{array} \right. \end{aligned}$$

Note that if  $b_j \preceq b_{j'}$ , then the constraint among events and transitions immediately ensures  $\text{dob}(b_{j,i}) \leq \text{dod}(b_{j,i}) \leq \text{dob}(b_{j',i}) \leq \text{dod}(b_{j',i})$ . However, we need to add a consistency constraint for every pair of concurrent conditions  $b_{i,j}, b_{i,j'}$  that belong to the same  $B_i$ . Hence, calling  $I(b_{i,j}, E, B)$  the set of conditions that represent the same place as  $b_{i,j}$  and are concurrent with  $b_{i,j}$  in  $\langle E, B \rangle$ , we have to ensure the constraint  $\text{non-blocking}(b_{i,j}) = \bigwedge_{b_{i,j'} \in I(b_{i,j}, E, B)} \text{no-overlap}(b_{i,j}, b_{i,j'})$ . In words, condition  $b_{i,j}$  does not hold during the validity dates of any concurrent condition representing the same place. In particular, a time process of  $\mathcal{N}$  cannot contain two maximal conditions with the same place.

Let us now consider the constraints attached to events. An event  $e = \langle B, t \rangle$  is an occurrence of a firing of transition  $t$  that needs conditions in  $B$  to be fulfilled to become enabled. Calling  $\text{doe}(e)$  the date of enabling of  $e$ , we necessarily have  $\text{doe}(e) = \max\{\text{dob}(b) \mid b \in B\}$ . Event  $e$  is fireable at least  $\text{eft}(t)$  time units, and at most  $\text{lft}(t)$  time units after being enabled. We hence have  $\text{doe}(e) + \text{eft}(t) \leq \text{dof}(e) \leq \text{doe}(e) + \text{lft}(t)$ . However, execution of  $e$  does not always occur immediately when  $e$  is fireable. Execution of  $e$  occurs after  $e$  is fireable, as soon as the places filled by  $e$  are empty, i.e.,  $e$  occurs at a date  $\theta(e)$  that guarantees that no place in  $t^\bullet$  is occupied. This is guaranteed by attaching to every event  $e$  the constraints  $\theta(e) = \text{dob}(b_1), \theta(e) = \text{dob}(b_2), \dots, \theta(e) = \text{dob}(b_k)$ , where  $\{b_1, b_2, \dots, b_k\} = e^\bullet$ , and constraints  $\text{non-blocking}(b_1), \text{non-blocking}(b_2), \dots, \text{non-blocking}(b_k)$ . Last, as semantics of STPNs is urgent, once fireable,  $e$  has to fire at the earliest possible date. This is encoded by the constraint  $\theta(e) = \min\{x \in \mathbb{R}_{\geq 0} \mid x \notin ]\text{dob}(b), \text{dod}(b)[ \text{ for some } b \in \bigcup I(b_i) \wedge x \geq \text{dof}(e)\}$ . Figure D.10 shows the effect of blocking and possible free firing dates for some event with a condition  $b$  in its postset. The top of the figure is a part of a pre-process, with conditions  $b, b_0, b_1, b_2$  referring to the same place  $p_1$ . Suppose that values of variables  $[\text{dob}(b_i) \text{ and } \text{dod}(b_i)]$  for  $i \in 0, 1, 2$  are already known. The situation is depicted by the drawing at the bottom of Figure D.10. Horizontal lines represent real lines, and line portion between brackets represent intervals  $[\text{dob}(b_i), \text{dod}(b_i)]$  for  $i \in 0, 1, 2$ . According to the considered pre-process, we have  $I(b) = \{b_0, b_1, b_2\}$ . Then  $[\text{dob}(b), \text{dod}(b)]$  have to be fully inscribed in one of these thick segments of the Figure. An event with  $b$  in its postset (as event  $e$  in the pre-process at the top of the Figure) can occur only at dates contained in these thick segments.

Written differently,

$$\theta(e) = \begin{cases} \text{dof}(e) & \text{if } \bigwedge_{b \in I(b_1) \cup \dots \cup I(b_k)} \text{dof}(e) \leq \text{dob}(b), \text{ and} \\ \min\{\text{dod}(b) \mid \forall b' \in \bigcup_{b_i \in e^\bullet} I(b_i), \text{dod}(b) \notin ]\text{dob}(b'), \text{dod}(b')[\} & \text{otherwise.} \end{cases}$$

This formula can be translated in boolean combinations of inequalities over variables of  $\text{var}(E, B)$ . Similarly, event  $e = \langle B, t \rangle$  must occur before all its conflicting events. If an event  $e'$  in conflict with  $e$  is executed, at least one condition in  $B$  is consumed, and  $e$  cannot occur in a time process containing  $e'$ . We hence need the additional constraint  $\bigwedge_{e' \# e} \text{notMoreUrg}(e, e')$  to guarantee that there exists no other event that is forced to occur before  $e$  due to urgency. We define  $\text{notMoreUrg}(e, e')$  as the following constraint:  
 $\text{notMoreUrg}(e, e') = \theta(e) \geq \text{doe}(e') + \text{lft}(\text{tr}(e')) \Rightarrow \text{tiled}(e, e') \vee \bigvee_{e'' \parallel e} \text{preempts}(e', e'')$

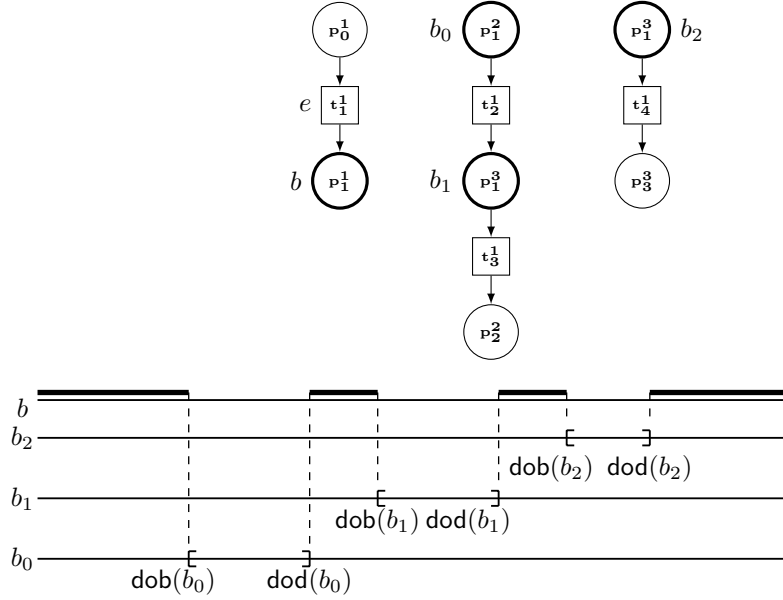


Figure D.10: Constraints on dates of birth of tokens in a shared place.

where  $\text{tiled}(e, e') = \text{free}(e') \cap [\text{doe}(e') + \text{lft}(\text{tr}(e')), \theta(e)] = \emptyset$ ,  $e'' || e$  refers to events that are concurrent with  $e$  in the considered set of events  $E$ ,  $\text{free}(e') = \mathbb{R}_{\geq 0} \setminus \{[\text{dob}(b), \text{dod}(b)] \mid \exists b' \in e'^{\bullet}, b \in I(b')\}$  is the set of intervals in which places attached to conditions in  $e'^{\bullet}$  are empty, and  $\text{preempts}(e', e'') = \theta(e'') \leq \min([\text{doe}(e') + \text{lft}(\text{tr}(e')), \theta(e)] \cap \text{free}(e''))$  means  $e''$  disabled  $e'$  by consuming a condition in  $e'^{\bullet}$ .

Constraint  $\text{notMoreUrg}(e, e')$  means that if  $e'$  is in conflict with  $e$ , then at least one condition in  $e'^{\bullet}$  is consumed before  $e'$  can fire, or if  $e'$  becomes firable before  $e$  fires, the urgent firing of  $e'$  is delayed by blockings so that  $e$  can occur. As for constraint attached to blockings,  $\text{notMoreUrg}(e, e')$  can be expressed as a boolean combination of inequalities. One can also notice that  $\text{notMoreUrg}(e, e')$  can be expressed without referring to variables attached to event  $e'$  nor  $e'^{\bullet}$ , as  $\text{doe}(e') = \max_{b_i \in e'^{\bullet}} \text{dob}(b_i)$  and the intersection of  $I(b)$  and  $e'^{\bullet}$  is void.

For causally closed sets of events and conditions  $E \cup B$  contained in some pre-process of  $\mathcal{U}_K$ , the constraint  $\Phi_{E,B}$  applying on events and conditions of  $E \cup B$  is now defined as  $\Phi_{E,B} = \bigwedge_{x \in E \cup B} \Phi_{E,B}(x)$  where:

$$\forall b \in B, \Phi_{E,B}(b) = \text{non-blocking}(b) \wedge \begin{cases} \text{dob}(b) = d_0 \text{ if } b \in B_0, \text{ and } b \text{ is maximal,} \\ \text{dob}(b) = d_0 \wedge \text{dob}(b) \leq \text{dod}(b) \text{ if } b \in B_0, \\ \text{dob}(b) = \theta(\bullet b) \text{ if } b \notin B_0 \text{ and } b \text{ is maximal,} \\ \text{dob}(b) = \theta(\bullet b) \wedge \text{dob}(b) \leq \text{dod}(b) \text{ otherwise.} \end{cases}$$

$$\forall e \in E, \Phi_{E,B}(e) = \begin{cases} \text{doe}(e) = \max_{b \in \bullet_e} \text{dob}(b) \\ \wedge \text{doe}(e) + \text{eft}(\text{tr}(e)) \leq \text{dof}(e) \leq \text{doe}(e) + \text{lft}(\text{tr}(e)) \\ \wedge \text{dof}(e) \leq \theta(e) \wedge \bigwedge_{b \in \bullet_e} \text{dob}(b) = \theta(e) \\ \wedge \bigwedge_{b \in e} \theta(e) = \text{dob}(b) \\ \wedge \bigwedge_{e' \# e} \text{notMoreUrg}(e, e') \end{cases}$$

Let us now address maximality of symbolic prefixes wrt urgent events firing. Let  $SPP = \langle E', B', \Phi_{E',B'} \rangle$  be a symbolic prefix of pre-process  $PP = \langle E, B \rangle$ . Symbolic process  $SPP$  is *maximal w.r.t urgent events firing* iff no more event of  $PP$  **have to** belong to  $SPP$ . This property of  $SPP$  holds if every event  $f \in B'^\bullet \cap E$  that could have become urgent before the last date of all events in  $E'$  was prevented from firing due to blocking. This property prefixes can be verified as a property  $\Phi_{\max}(f)$  that have to be satisfied for every  $f \in B'^\bullet \cap E$ . Let  $C_f = \text{pl}^{-1}(f^\bullet) \cap B'$  denote the set of conditions of  $B'$  whose place appears in the postset of  $f$ . Then,  $SPP$  is maximal iff for every  $f \in B'^\bullet \cap E$ , the following constraint is not satisfiable.

$$\Phi_{\max}(f) = \begin{cases} \Phi_{E',B'} \\ \wedge \theta(f) \leq \max_{e' \in E'} \theta(e') \text{ (f fires before the last event in } E') \\ \wedge \text{eft}(f) + \max_{b \in \bullet_f} \text{dob}(b) \leq \theta(f) \text{ (f is urgent)} \\ \wedge \bigvee_{X \in 2^{C_f}} \max_{x \in X} \text{dob}(x) \leq \theta(f) \leq \min_{x \in C_f \setminus X} \text{dob}(x) \\ \text{(f is not blocked for the whole duration of the process)} \end{cases}$$

Intuitively,  $\Phi_{\max}(f)$  means that  $f$ , that is not in the symbolic process, becomes urgent, is not blocked by conditions in  $B'$ , and has to fire before the execution of the last event in  $E'$ . If  $\Phi_{\max}(f)$  is satisfiable, then  $f$  should appear in the process.

## Appendix E. Proof of proposition 1

**Proposition 1.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  time units (between consecutive occurrences of each transition). For every date  $D \in \mathbb{R}_{\geq 0}$  and condition  $b$  in an unfolding  $\mathcal{U}_n$ , there exists  $K \geq n$  s.t.  $\{b' \in \text{block}(b) \mid \text{dob}(b') \leq D\}$  is contained in  $\mathcal{U}_K$ .

**Proof:** Consider a pre-process  $PP$  of  $\mathcal{U}_n$ , which depth is more than  $\lceil \frac{D}{\delta} \rceil \cdot |T|$  events. Every event of the unfolding appended at depth  $i$  consumes conditions that were created at depth  $j < i$ , and at least one condition that was produced at step  $i$  of the unfolding. Hence, for every event  $e_n$  and  $b_n$  condition created at depth  $n$ , there exists a sequence  $b_0 < e_1 < b_1 < \dots < e_n < b_n$  of events and conditions of increasing depth (and also increasing dates). With the time progress assumption, we know that every consecutive pair of events representing the same transition occurs at least at dates that differ by  $\delta$ . Hence, an event created at depth  $n$  has an occurrence date of at least  $\delta \cdot \lfloor n/|T| \rfloor$ . The occurrence date of an event created at depth greater than  $\frac{D}{\delta} \cdot |T|$  is hence greater than  $D$ . The number of events and conditions created at step  $n$  and appearing in the same pre-process of  $\mathcal{U}_n$  is finite (as creating an event uses exclusively at least one condition of the preceding step). It is hence sufficient to unfold a net up to

1305 depth  $\frac{D}{\delta} \cdot |T|$  to obtain the (finite) set of conditions that refer to the same place  
as some condition  $b$  before a given date  $D$ .  $\square$

## Appendix F. Proof of Proposition 2

**Proposition 2.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  time units.  
The set of time processes executable by  $\mathcal{N}$  in  $D$  time units are prefixes of time  
1310 processes of  $\mathcal{U}_K$ , with  $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$  containing only events with date  $\leq D$ .

**Proof:** We will show inclusion of the set of processes in the two directions.  
First of all, we define an ordering on symbolic processes. Let  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  and  
 $\mathcal{E}^{s'} = \langle E', B', \Phi' \rangle$  be two symbolic processes. We will say that  $\mathcal{E}^s \sqsubseteq \mathcal{E}^{s'}$  iff there  
exists an event  $e'$  such that  $E' = E \cup \{e'\}$ ,  $B' = B \cup e'^\bullet$ , and  $\Phi = \Phi'_{|\text{var}(E, B)}$ .  $\square$

1315 **Lemma 1.** Let  $\mathcal{E}^s$  be a symbolic process of unfolding  $\mathcal{U}_K$ , starting from  $m_0, d_0$ ,  
that is satisfiable and complete. Let  $\theta$  be one of its solutions guaranteeing  
 $\forall e \in E, \theta(e) \leq D$ . Then, there exists a sequence  $\mathcal{E}^{s,0} = \langle E_0, B_0, \Phi_0 \rangle \sqsubseteq \mathcal{E}^{s,1} =$   
 $\langle E_1, B_1, \Phi_1 \rangle \cdots \sqsubseteq \mathcal{E}^s$  of symbolic processes of  $\mathcal{U}_K$  such that  $E_0 = \emptyset$ ,  $B_0 =$   
 $\{\langle \perp, p \rangle \mid p \in m_0\}$   $\Phi_0 = \{\theta(\perp) = d_0 \wedge \bigwedge_{b \in B_0} \text{dob}(b) = d_0\}$  and  $\theta$  is a solution for  
1320 every  $\mathcal{E}^{s,i}$  and  $\theta(e_i) \leq \theta(e_i + 1)$ .

**Proof:** We can show this property by induction on the size of prefixes of  
 $\mathcal{E}^s$ . The base hypothesis is straightforward, taking the sequence with only one  
symbolic process  $\mathcal{E}^{s,0}$  without events. Suppose that this property is satisfied  
for symbolic processes up to size  $n$ , and consider a satisfiable and complete  
1325 symbolic process  $\mathcal{E}^{s,n+1}$  of size  $n+1$ . Let  $\theta_{n+1}$  denote a solution for this process.  
A growing sequence from  $\mathcal{E}^{s,0}$  to  $\mathcal{E}^{s,n+1}$  exists. In this sequence, the difference  
between  $\mathcal{E}^{s,n+1}$  and  $\mathcal{E}^{s,n}$  is a single event  $e$  that is maximal in  $\mathcal{E}^{s,n+1}$  w.r.t.  
ordering on events  $\preceq$ , and such that  $\theta(e) \geq \theta(x)$  for every event  $x$  in  $\mathcal{E}^{s,n+1} \setminus \{e\}$ ,  
and  $\theta(e) \geq \text{dob}(b)$  for every  $b$  in  $\mathcal{E}^{s,n+1} \setminus \{e\}$ . Let  $E^n, B^n$  denote the set of events  
1330 in  $\mathcal{E}^{s,n+1} \setminus \{e\}$ . Let us denote by  $\Phi_{n+1|E^n, B^n}$  the restriction of  $\Phi_{n+1}$  to variables  
attached to events and conditions  $E^n, B^n$ . One has to remove variables  $\theta(e)$ ,  
 $\text{dod}(b)$  for every  $b \in e^\bullet$ , and  $\text{dob}(b)$  for every  $b \in e^\bullet$  using an elimination technique  
such as Fourier-Motzkin. Using the properties of elimination,  $\theta$  satisfies  $\Phi_{n+1}$   
if and only if the restriction of  $\theta$  to  $\text{var}(E^n, B^n)$  satisfies  $\Phi_{n+1|E^n, B^n}$ . However,  
1335 the restriction of  $\theta$  is exactly  $\theta_n$ , and as  $\theta(e)$ ,  $\text{dod}(b)$  for  $b \in e^\bullet$ , and  $\text{dob}(b)$  for  
 $b \in e^\bullet$  are all greater than variables in  $\text{var}(E^n, B^n)$ , the elimination of variables  
is simply a projection on atoms that do not contain variables related to  $e$ , and  
 $\Phi_{n+1|E^n, B^n} = \Phi_n$ .  $\square$

**Lemma 2.** Given a symbolic process  $\mathcal{E}^s$  of  $\mathcal{U}_K$ , one of its solutions  $\theta$ , and an  
1340 ordering  $\mathcal{E}^{s,0} = \langle E_0, B_0, \Phi_0 \rangle \sqsubseteq \mathcal{E}^{s,1} = \langle E_1, B_1, \Phi_1 \rangle \cdots \sqsubseteq \mathcal{E}^s$  as above, then the  
word  $u_{\mathcal{E}^s, \theta} = \langle t_1, \theta(e_1) \rangle \cdots \langle t_{|E|}, \theta(e_{|E|}) \rangle$  is a timed word of  $\mathcal{L}(\mathcal{N})$ .

**Proof:** Again we can prove this lemma by induction. The base case is obvious, as the empty word  $\epsilon$  is a timed word of  $\mathcal{L}(\mathcal{N})$ . Let us suppose that the property is satisfied up to  $n$ , that is for every process  $\mathcal{E}_n$  of size  $n$  and solution  $\theta_n$  meeting all constraints of  $\mathcal{E}_n$ , there exists an increasing sequence of prefixes of  $\mathcal{E}_n$  such that the word associated with this sequence is a timed word of  $\mathcal{L}(\mathcal{N})$ .

Let us now consider a time process  $\mathcal{E}_{n+1}$  with  $n+1$  events and one of its solutions  $\theta_{n+1}$ . As in Lemma 1, one can find an event  $e_{n+1}$  and a process  $\mathcal{E}_n$  such that  $\mathcal{E}_n$  and  $\mathcal{E}_{n+1}$  only differ by addition of this single event. There exists a timed word  $u_n = \langle e_1, \theta_{n+1}(e_1) \rangle \dots \langle e_n, \theta_{n+1}(e_n) \rangle \in \mathcal{L}(\mathcal{N})$  corresponding to  $\mathcal{E}_n$ . This word may lead the net to any configurations in a set  $Conf_n$  with identical markings, but distinct times to fire attached to transitions. However, as we know that  $\theta_{n+1}$  meets all constraints of  $\mathcal{E}_{n+1}$ , there exists a configuration in  $Conf_n$  whose times to fire allow firing of  $e_{n+1}$  at date  $\theta(e_{n+1})$ , and  $u_{n+1} = u_n \cdot \langle e_{n+1}, \theta(e_{n+1}) \rangle \in \mathcal{L}(\mathcal{N})$ .  $\square$

Note that assuming time progress, the dates attached to an event of a process of  $\mathcal{U}_K$  that occur at a date smaller than  $D$  cannot be further constrained by addition of constraints coming from events that are not in  $\mathcal{U}_K$ . The two lemmas above hence allow to conclude that for a given symbolic process  $\mathcal{E}^s$  of unfolding  $\mathcal{U}_K$ , in which one considers events that occur before date  $D$ , and for each solution of  $\mathcal{E}^s$ , we have  $\mathcal{E}_\theta^s = TP_{u_{\mathcal{E}^s, \theta}}$  for some word  $u_{\mathcal{E}^s, \theta} \in \mathcal{L}(\mathcal{N})$ . Hence, the set of time processes of  $\mathcal{U}_K$  whose events occur before  $D$  is contained in the set of time processes  $TP(\mathcal{L}_{\leq D}(\mathcal{N}))$ . All time processes of some pre-process of  $\mathcal{U}_K$  (and hence all time processes of unfolding  $\mathcal{U}_K$ ) can be built from a timed word that is executable by  $\mathcal{N}$  in less than  $D$  time units, and are hence time processes of  $\mathcal{N}$ .

We now have to prove the converse direction, i.e., every time process associated with a word  $u \in \mathcal{L}_{\leq D}(\mathcal{N})$  is a time process of  $\mathcal{U}_K$ .

**Lemma 3.** *Let  $u \in \mathcal{L}_{\leq D}(\mathcal{N})$ . Then,  $TP(u)$  is a time process of  $\mathcal{U}_K$ .*

**Proof:** We proceed by induction on the size of words. First, for the empty words, the time process with only initial conditions is clearly a time process of  $\mathcal{U}_K$ . Let us now assume that for every  $u_n = \langle e_1, \theta(e_1) \rangle \dots \langle e_n, \theta(e_n) \rangle \in \mathcal{L}_{\leq D}(\mathcal{N})$  of length  $n$ ,  $TP(u_n)$  is a time process of  $\mathcal{U}_K$ . Let us consider a word  $u_{n+1} = \langle e_1, \theta(e_1) \rangle \dots \langle e_n, \theta(e_n) \rangle \cdot \langle e_{n+1}, \theta(e_{n+1}) \rangle \in \mathcal{L}_{\leq D}(\mathcal{N})$ . One can build a time process  $\mathcal{E}_u$  for  $u = \langle e_1, \theta(e_1) \rangle \dots \langle e_n, \theta(e_n) \rangle$ . Clearly, as  $u_{n+1} \in \mathcal{L}_{\leq D}(\mathcal{N})$ , word  $u$  leads from marking  $m_0$  to a marking that enables  $e_{n+1}$ . Let  $e_{p_1}, \dots, e_{p_k}$  denote the  $k$  events that produce the tokens that are consumed by  $e_{n+1}$ . If event  $e_{n+1}$  is a firing of some transition  $t$  that occurs exactly when its time to fire has expired,  $\theta(e_{n+1})$  meets the constraint  $\text{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1}) \leq \text{lft}(t) + \max\{\theta(e_{p_i})\}$ . In any case, we have  $\text{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1})$  (which is the only constraint w.r.t predecessors imposed by constraint in the unfolding. Similarly, let  $e_{b_1}, \dots, e_{b_q}$  denote the last events of  $u$  that free places in which  $t$  outputs some tokens (and hence may have blocked the execution of  $t$  before  $\theta(e_{n+1})$ ). We have  $\theta(e_{n+1})$  meets the constraint  $\max\{\theta(e_{b_i})\} \leq \theta(e_{n+1})$ . Hence, any event that had to occur before  $\theta(e_{n+1})$  (due to urgency, causality, or blockings) also appears in  $\mathcal{E}_u$ . Hence,  $\theta$  witnesses satisfiability of a set of

constraints over occurrence dates of events  $e_1, \dots, e_n$ , and one can safely append  $e_{n+1} = (B, t)$  to maximal places of  $\mathcal{E}_u$ , and obtain a symbolic prefix  $\mathcal{E}_{u_{n+1}}$  (satisfiable, conflict free and complete). It now remains to show that  $\mathcal{E}_{u_{n+1}}$  is a symbolic process of  $\mathcal{U}_K$ . As  $\theta(e_{n+1}) \leq D$ ,  $e_{n+1}$  appears in the unfolding of  $\mathcal{N}$  at depth at most  $\frac{D}{\delta}$ , which is lower than  $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$ . Hence,  $\mathcal{E}_{u_{n+1}}$  is an causally closed set of events that also contains all mandatory urgent transition firings and place unblockings whose set of constraints is satisfiable, and contained in  $\mathcal{U}_K$ , i.e., it is a symbolic process of  $\mathcal{U}_K$ .  $\square$

## Appendix G. Algorithm to compute embeddings of a schedule in a process

Finding the set of embedding functions  $\Psi = \{\psi_0, \psi_1, \dots, \psi_k\}$  that satisfy the conditions of definition 14 is achieved building iteratively injective functions meeting the embedding requirements, by matching at every step minimal yet unexplored nodes of  $S$  with minimal unmatched nodes of  $\mathcal{E}^s$ . This construction is depicted in Algorithm 1. We will define an embedding function  $f$  as a set pairs of the form  $\langle n, e \rangle$ , interpreted as  $f(n) = e$ . We define in particular the empty embedding  $f_\perp$ , that is undefined for every node of  $N$ , and will be used as starting point of the algorithm. For a given function  $f$ , the function  $f \cup \langle n, e \rangle$  is the function that associates  $e$  to node  $n$ , and  $f(n')$  to every other node  $n' \in \text{domain}(f)$ .

## Appendix H. Stochastic state class tree

In this part of the appendix, we detail how to build a stochastic state class tree for a particular process of a stochastic Petri net with blocking semantics.

**Definition 17 (transient stochastic state class).** A transient stochastic state class (or class for short) of an STPN  $\mathcal{N}$  is a tuple  $\Sigma = \langle m, D, f_\tau, \text{BLK}, \text{URG} \rangle$  where  $m$  is a marking,  $\tau = \langle \tau_{\text{age}}, \underline{\tau} \rangle$  is a vector where  $\tau_{\text{age}}$  is the opposite of the elapsed time and  $D$  is a domain for  $\tau$ .  $\underline{\tau} = \langle \tau_0, \tau_1, \dots, \tau_{N-1} \rangle$  is a vector of random variables with support  $D_{\downarrow \tau_{\text{age}}}$  representing the possible TTFs of transitions enabled by marking  $m$  s.t. for every  $\tau_i$  in  $\underline{\tau}$ , the elimination of all other variables from  $D$  yields a non-empty set of possible values that is different from  $\{0\}$ ;  $f_\tau$  is a PDF over  $D$ ,  $\text{BLK}$  is a set of blocked transitions, and  $\text{URG}$  is a set of urgent transitions.

The domain  $D$  of  $\tau$  is simply the domain of a class as defined in the usual state class graph construction of TPNs (see for instance [19, 18]). It represents possible values for TTFs attached to transitions. As our STPN is bounded, the number of domains that can be generated inductively at construction time is finite (as proved by [18]). Urgent and blocked transitions are also finite subsets of  $T$ . However, as shown in [3], the number of distributions that can be iteratively computed needs not be finite.

---

**Algorithm 1:** Computation of embeddings of a schedule in a symbolic process

---

**input:** a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ , a symb. process  $\mathcal{E} = \langle E, B, \Phi \rangle$ ;

$\Psi := \emptyset$ ;                                   // the set of solutions is initially empty

$F := \{f_\perp\}$ ;                           // the exploration starts from the undefined map

**while**  $F \neq \emptyset$  **do**

    choose  $f \in F$ ;

$\text{MIN}_{S,f} := \min_{\rightarrow} (N \setminus \text{domain}(f))$ ;

**if**  $\text{MIN}_{S,f} = \emptyset$  **then** ;       // all nodes of  $S$  have an image in  $\mathcal{E}$

$\Psi := \Psi \cup \{f\}$ ;                                   //  $f$  is an embedding

**else**

$F := F \setminus \{f\}$ ;       // we will explore extensions of partial embedding  $f$

$\text{MIN}_{\mathcal{E},f} := \min_{\preceq} (E \setminus \text{image}(f))$ ;

        FOUND := false;

**while**  $\text{MIN}_{S,f} \neq \emptyset \wedge \text{FOUND} = \text{false}$  **do**

            choose  $n \in \text{MIN}_{S,f}$ ;

$\text{MIN}_{S,f} := \text{MIN}_{S,f} \setminus \{n\}$ ;

$\text{CAND} := \{e \in \text{MIN}_{\mathcal{E},f} \mid \text{tr}(e) \in r(\lambda(n)) \wedge \forall n' \in \text{dpred}(n), f(n') \preceq_{\rightarrow} e\}$ ;

**if**  $\text{CAND} \neq \emptyset$  **then**

$F := F \cup \bigcup_{e \in \text{CAND}} \{f \cup \langle n, e \rangle\}$ ;   // update  $f$  with pairs  $\langle n, e \rangle$  that meet the matching criteria and add the new functions to candidate embeddings

                FOUND := true;

**end**

**end**

**end**

**end**

---



1425 **Definition 18 (transient stochastic state class tree).** A transient stochastic state class tree for an STPN  $\mathcal{N}$  (that we shall call *tree*, for short) is a directed acyclic graph  $\mathfrak{S} = \langle V, \circ \rightarrow \cup \bullet \rightarrow \rangle$  where vertices in  $V$  are classes, edges in  $\circ \rightarrow$  represent firing transitions after (symbolically) elapsing time, and edges in  $\bullet \rightarrow$  represent firings of urgent transitions. Every vertex in the tree has only one predecessor except for the root of the tree, denoted  $v_0$ , that has no predecessor. Edges carry probabilistic informations on transitions firings and the sum of probabilities of all edges leaving the same vertex is equal to 1.

The construction of a tree starts from the initial class  $\Sigma_0$  (with marking  $m_0$ , a domain  $D_0$  for the TTFs of transitions enabled in  $m_0$  and all other components defined accordingly, see appendix E) and inductively computes edges and reachable classes. Edges  $\Sigma \xrightarrow{t, \mu} \Sigma'$  from a class  $\Sigma$  to a successor class  $\Sigma'$  are labeled by a transition name  $t$  and by the probability  $\mu$  to fire  $t$  from  $\Sigma$ , and are of two forms:

**Firing after elapsing time:** A move  $\Sigma \xrightarrow{t_i, \mu_i} \Sigma'$  from  $\Sigma = \langle m, D, f_{\mathcal{T}}, \text{BLK}, \text{URG} \rangle$  to  $\Sigma' = \langle m', D', f_{\mathcal{T}'}, \text{BLK}', \text{URG}' \rangle$ , achievable with probability  $\mu_i$ , consists in firing transition  $t_i$  after symbolically elapsing its TTF. Such a move is only allowed if  $\text{URG} = \emptyset$  and the TTF  $\tau_i$  of  $t_i$  is less than or equal to TTFs of all other transitions that could fire from  $\Sigma$ . The time domain  $D^i$  from which  $t_i$  can fire is hence  $D^i = D \cap \bigcap_{\tau_j \in \mathcal{T}} \{\tau_i \leq \tau_j\}$ , and the probability of firing  $t_i$  from  $\Sigma$  is  $\mu_i = \int_{D^i} f_{\mathcal{T}}(x) dx$ . We have  $m' = m - \bullet t_i + t_i \bullet$ . The new domain  $D'$  and distribution  $f_{\mathcal{T}'}$  are computed as for STPNs with non-blocking semantics: Vector  $\tau$  is obtained by advancing time, removing variables of disabled transitions and adding those of newly enabled transitions [3], and then removing variables of transitions whose domain is the singleton  $\{0\}$ . The domain of a single variable  $\tau_i$  in a domain  $D$  over several variables can be obtained by eliminating all variables but  $\tau_i$  from  $D$ . As soon as a variable has domain  $\{0\}$ , the time to fire of the associated transition is necessarily zero, and the transition has to fire. It is then stored in the set of urgent transitions if it is not blocked, and in the set of blocked transitions otherwise. The set  $\text{BLK}'$  is obtained by removing from  $\text{BLK}$  transitions that were disabled by firing of  $t_i$  and transitions that are not blocked anymore in  $m'$  thanks to the places freed by firing of  $t_i$  (they become urgent), and adding transitions which are enabled in  $m'$  with a firing domain in  $D'$  that is  $\{0\}$ . Finally, the set  $\text{URG}'$  contains all enabled transitions that became urgent when firing  $t_i$ , i.e., transitions with firing domain  $\{0\}$  among enabled transitions, and formerly blocked transitions unblocked by  $t_i$ .

**Firing urgent transitions:** In STPNs semantics, when more than one transition is fireable from a configuration, their weights are used to compute the probability of firing each transition. This case can occur because of blocking semantics: an STPN can keep several transitions blocked, and firing a transition can also unlock several of them at the same time (all unblocked transitions become urgent). When a class  $\Sigma$  has urgent transitions ( $\text{URG} \neq \emptyset$ ), only moves of the form  $\Sigma \xrightarrow{t_i, \mu_i} \Sigma'$  are allowed. They consist in firing a transition  $t_i$  among urgent transitions in  $\text{URG}$  with probability  $\mu_i = \mathcal{W}(t_i) / \sum_{t_j \in \text{URG}} \mathcal{W}(t_j)$ . Components

1470  $m'$ ,  $D'$  and  $f_{\tau}'$  of the successor class are computed as for timed moves, with the only difference that no time elapses before the firing of  $t_i$ . Set  $\text{BLK}'$  is obtained by removing from  $\text{BLK}$  transitions that are unlocked or disabled by the firing of  $t_i$  and adding those that become blocked in  $m'$ , and  $\text{URG}'$  contains transitions from  $\text{URG}$  that were not disabled by firing of  $t_i$  and transitions from  $\text{BLK}$  that were unblocked when firing  $t_i$ .

1475 Transient trees are a priori infinite, but very often, one can work with a bounded horizon. This is our case when evaluating the probability of realization in  $\mathcal{U}_K$ . Let  $\Psi_i = \{\psi_{i,0}, \psi_{i,1}, \dots, \psi_{i,n-1}\}$  denote all possible embeddings of a schedule  $S$  into a symbolic process  $\mathcal{E}_i^s$  of  $\mathcal{N}$ . We denote by  $\mathbb{P}(\Phi_{\psi_{i,j}, d \pm \alpha})$  the probability that  $\mathcal{N}$  executes a time process  $\mathcal{E}_i^s$  and realizes  $S$  within a precision of  $\pm \alpha$  when  $\psi_{i,j}$  is the embedding of  $S$  in  $\mathcal{E}_i^s$ . We adapt the tree construction to consider only  $\mathcal{E}_i^s$  and embedding  $\psi_{i,j}$ , and compute  $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, d \pm \alpha})$ . We build a tree whose vertices of the form  $\langle \Sigma, S \rangle$  memorize a class and a suffix of  $\mathcal{E}_i^s$  not yet executed. We start from vertex  $\langle \Sigma_0, \mathcal{E}_i^s \rangle$ . We create an edge  $\langle \Sigma, S \rangle \xrightarrow{t_k, \mu_k} \langle \Sigma', S' \rangle$  representing firing of a transition  $t_k$  if there is a minimal event  $e$  in the remaining suffix of  $\mathcal{E}_i^s$ , and  $\text{tr}(e) = t_k$ .  $S'$  is the new suffix obtained by removing  $e$  from  $S$ .  $\Sigma'$  is the successor class obtained after firing this transition from  $\Sigma$ . Edges are built as before in the tree, but with an additional constraint: edges with label  $t_k, \mu_k$  and components  $m, D, f_{\tau}, \text{BLK}, \text{URG}$  of classes are built with the additional requirement that when creating an edge from an event  $e$  that is in the image of  $\psi_{i,j}$ , the firing time domain is restricted to impose that  $e$  occurs in the time interval  $I_k = [\max(0, d(\psi_{i,j}^{-1}(e)) - \alpha), d(\psi_{i,j}^{-1}(e)) + \alpha]$ . In this case, the probability of firing  $t_k = \text{tr}(e)$  becomes  $\mu_k = \int_{D^k \cap D'^k} f_{\tau}(x) dx$ , where  $D'^k$  is the part of  $D$  in which  $\tau_k - \tau_{\text{age}}$  (the firing date for  $e$ ) belongs to  $I_k$ . We stop developing a branch of the tree at vertices whose suffix does not contain events that are images of nodes in  $S$  via  $\psi_{i,j}$ . The construction ends with a tree  $\mathfrak{S}_{i,j,\alpha}$ .

1495 Transient tree  $\mathfrak{S}_{i,j,\alpha}$  measures the probability of solutions and of occurrence of a particular process. The probability  $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha})$  is computed as  $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha}) = \sum_{\rho \in \text{PATH}(\mathfrak{S}_{i,j,\alpha})} \mathbb{P}(\rho)$  where  $\text{PATH}(\mathfrak{S}_{i,j,\alpha})$  is the set of paths from  $\langle \Sigma_0, \mathcal{E}_i^s \rangle$  to a leaf of  $\mathfrak{S}_{i,j,\alpha}$ , and the probability  $\mathbb{P}(\rho)$  of a path  $\rho = \Sigma_0 \xrightarrow{t_i, \mu_i} \dots \xrightarrow{t_{l-1}, \mu_{l-1}} \Sigma_l$  that start at  $\Sigma_0$  and end on a leaf  $\Sigma_l$  is the product  $\prod_{i \in \{0, \dots, l-1\}} \mu_i$ . As soon as  $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha}) > 0$ ,  $S$  has a non-null probability to be realized (with a tolerance of  $\pm \alpha$ ). Noticing that different embeddings yield disjoint paths in their respective transient stochastic state class trees, the probability for a schedule to be realized by a process is hence

1500  $\mathbb{P}(\mathcal{E}_i^s \models S) = \sum_{\psi_{i,j} \in \Psi_i} \mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha})$ .

1505 Finally, denoting by  $\mathcal{E}(PP, S)$  the symbolic processes of a pre-process  $PP$  that embed  $S$ , the probability  $\mathbb{P}(\mathcal{N} \models S)$  is greater than  $\max\{\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha}) \mid PP \in \mathcal{PE}(U_K) \wedge \mathcal{E}_i^s \in \mathcal{E}(PP, S)\}$ . It is difficult to obtain more than a lower bound for realization, as symbolic processes of  $\mathcal{E}(PP, S)$  might have overlapping executions.

## Appendix I. Derivation of components of successor class

We hereafter provide details on how to compute components  $D'$  and  $f_{\tau'}$  of a class  $\Sigma'$  obtained from a class  $\Sigma$  through a transition  $\xrightarrow{t_i, \mu_i}$ . Derivation of components  $m'$ ,  $\text{BLK}'$  and  $\text{URG}'$  has already been covered and need not further explanations.

We shall use the following notations:

- given a time domain  $D$  delimiting the possible values of a set of variables  $\underline{x} = \langle x_0, x_1, \dots, x_{N-1} \rangle$ , we will denote by  $D \downarrow_{x_i}$  the projection of  $D$  that eliminates variable  $x_i$  from  $\underline{x}$ . The elimination is done via the Fourier-Motzkin method detailed in Appendix Appendix J;
- given a vector  $\underline{x} = \langle x_0, x_1, \dots, x_{N-1} \rangle$ , we denote by  $\underline{x} \setminus x_i$  the vector  $\underline{x}$  from which variable  $x_i$  is removed, with  $i \in \{0, 1, \dots, N-1\}$ ;
- the addition of a scalar  $x_0$  to each element of a vector  $\underline{x} = \langle x_1, x_2, \dots, x_n \rangle$  is simply written  $\underline{x} + x_0$ .

Fist of all, the initial class is  $\Sigma_0 = \langle m_0, D_0, f_{\tau_0}, \text{blk}_0, \text{urg}_0 \rangle$ . Marking  $m_0$  is the initial marking of the net, domain  $D_0$  is a product of domains  $[eft(t), lft(t)]$  for all transitions enabled in  $m_0$  (their domain is not  $\{0\}$  as we have  $eft(t) < lft(t)$ ). Vector  $\tau_0 = \langle 0, \tau_1, \tau_{enab(m_0)} \rangle$  is a vector representing elapsed time (in the initial class it must be 0, and times to fires of enable variables. As domains for  $\tau_i$ 's are not singletons, we can safely set  $\text{blk}_0 = \emptyset$ , and  $\text{urg}_0 = \emptyset$ .

**Probability of firing:** A transition  $t_i$  can fire from class  $\Sigma$  iff  $t_i$  is enabled by  $m$  and no postset place of  $t_i$  is occupied; that is  $\forall p \in t_i^\bullet, m(p) = 0$ . We also need its TTF  $\tau^i$  to be less than or equal to TTFs of all variables in  $\tau$ ; transition  $t_i$  will then fire from  $\Sigma$  with probability  $\mu^i$ , with:

$$\mu^i = \int_{D^i} f_{\tau}(x) dx$$

$D^i$  is the time domain from which  $t_i$  shall fire with all its TTF less than or equal to every variable in  $\tau$ .

$$D^i = D \cap \bigcap_{t^j \in \tau} \{\tau^i \leq \tau^j\}$$

**Precedence condition:** The assumption that  $t_i$  fires before any other transition adds conditions on the time vector and thus leads to a new random variable  $\tau_a$  distributed over  $D^i$  according to the following conditional PDF:

$$f_{\tau_a}(x) = f_{\tau}(x) / \mu^i$$

**Time elapsing and elimination:** According to the semantics of STPNs, when  $t_i$  fires, TTFs of activated transitions are decreased by the value of the TTF of  $t_i$ , namely  $\tau^i$ . This yields a new random variable  $\tau_b = \tau_a - \tau^i$  distributed over

the domain  $D_b = D^i \downarrow_{\tau^i}$  in which the variable attached to the fired transition  $t_i$  is eliminated. The PDF of the new multivariate random variable  $\tau_b$  is then:

$$f_{\tau_b}(\underline{x}) = \int_{\text{MIN}^i}^{\text{MAX}^i} f_{\tau_a}(\underline{x} + x_i, x_i) dx_i$$

where  $\text{MIN}^i$  and  $\text{MAX}^i$  denote the bounds of the support of variable  $\tau^i$ .

**Disabling:** If the firing of  $t_i$  disables a transition  $t_j$ , variable  $\tau_b^j$  has to be eliminated from the time vector, yielding a new vector  $\tau_c = \tau_b \setminus \tau_b^j$  distributed over  $D_c = D_b \downarrow_{\tau_b^j}$  with PDF:

$$f_{\tau_c}(\underline{x}) = \int_{\text{MIN}^j}^{\text{MAX}^j} f_{\tau_b}(\underline{x}, x_j) dx_j$$

The same procedure is repeated for every disabled transition by the firing of  $t_i$ . Let  $\tau_{c^*}$ ,  $D_{c^*}$  and  $f_{\tau_{c^*}}$  then respectively denote the resulting time vector, domain and PDF.

**Newly enabling:** If the firing of  $t_i$  enables a transition  $t_k$ , with PDF  $f_{t_k}$  over  $[\text{eft}(t_k), \text{lft}(t_k)]$ , then the new time vector, that we denote by  $\tau_d$ , shall include an additional component  $\tau_d^k$  and shall be distributed over  $D_d = D_{c^*} \times [\text{eft}(t_k), \text{lft}(t_k)]$  according to the PDF:

$$f_{\tau_d}(\underline{x}, x_k) = f_{\tau_{c^*}}(\underline{x}) \times f_{t_k}(x_k)$$

1535 The same procedure is similarly repeated for every newly enabled transition to finally obtain the PDF of the successor class  $\Sigma'$ .

## Appendix J. Fourier–Motzkin elimination method

The *Fourier–Motzkin elimination method* is an algorithm for eliminating variables from a system of linear inequalities. Let  $\phi$  be a system of linear inequalities with variables  $x_1, x_2, \dots, x_r$  where  $x_r$  is the variable to be removed.

$\phi$  can be written as  $\phi^+ \wedge \phi^- \wedge \phi^\emptyset$  where  $\phi^- = \bigwedge_{i=1}^m -x_r \leq b_i - \sum_{k=1}^{r-1} a_{ik}x_k$  and

$\phi^+ = \bigwedge_{i=1}^n x_r \leq b_i - \sum_{k=1}^{r-1} a_{ik}x_k$  are the sets of inequations where the coefficients of

$x_r$  are respectively negative and positive, and  $\phi^\emptyset$  is the inequations subsystem in which  $x_r$  doesn't appear. Eliminating the variable  $x_r$  from  $\phi$  refers to the creation of another system of inequations in which  $x_r$  doesn't appear and which has the same solutions over the remaining variables. The original system can be written as

$$\max_{i=1, \dots, m} (-b_i + \sum_{k=1}^{r-1} a_{ik}x_k) \leq x_r \leq \min_{i=1, \dots, n} (b_i - \sum_{k=1}^{r-1} a_{ik}x_k) \wedge \phi^\emptyset$$

Which, by eliminating  $x_r$ , gives

$$\max_{i=1,\dots,m} (-b_i + \sum_{k=1}^{r-1} a_{ik}x_k) \leq \min_{i=1,\dots,n} (b_i - \sum_{k=1}^{r-1} a_{ik}x_k) \wedge \phi^\emptyset$$

**example:** Let  $\phi$  be the following system of linear inequalities:

$$\phi = \begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ \alpha_2 \leq x_2 \leq \beta_2 \\ \alpha_3 \leq x_3 \leq \beta_3 \\ -\gamma_{21} \leq x_1 - x_2 \leq \gamma_{12} \\ -\gamma_{31} \leq x_1 - x_3 \leq \gamma_{13} \\ -\gamma_{32} \leq x_2 - x_3 \leq \gamma_{23} \end{cases}$$

Suppose that we want to eliminate the variable  $x_1$ . We start by identifying  $\phi^\emptyset$ , the subsystem of inequations in which  $x_1$  doesn't appear, as follows:

$$\phi = \begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ -\gamma_{21} \leq x_1 - x_2 \leq \gamma_{12} \\ -\gamma_{31} \leq x_1 - x_3 \leq \gamma_{13} \\ \alpha_2 \leq x_2 \leq \beta_2 \\ \alpha_3 \leq x_3 \leq \beta_3 \\ -\gamma_{32} \leq x_2 - x_3 \leq \gamma_{23} \end{cases} = \phi^\emptyset$$

We then isolate  $x_1$  by rewriting the inequations left, as follows:

$$\phi = \begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ x_2 - \gamma_{21} \leq x_1 \leq x_2 + \gamma_{12} \\ x_3 - \gamma_{31} \leq x_1 \leq x_3 + \gamma_{13} \\ \phi^\emptyset \end{cases}$$

One can easily see that an equivalent solution of this system is the one where  $x_1$  is bounded with the maximum value of its left bounds and the minimum of its right bounds in the system of inequations; adding to that  $\phi^\emptyset$ .

$$\phi \iff \max(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}) \leq x_1 \leq \min(\beta_1, x_2 + \gamma_{12}, x_3 + \gamma_{13}) \wedge \phi^\emptyset$$

Finally, eliminating  $x_1$  consists in saying that the left bound is inferior or equal to the right bound and we get the following:

$$\phi' \iff \max(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}) \leq \min(\beta_1, x_2 + \gamma_{12}, x_3 + \gamma_{13}) \wedge \phi^\emptyset$$